

XACML 3.0 in Answer Set Programming – Extended Version

Caroline Dewi Puspa Kencana Ramli, Hanne Riis Nielson, Flemming Nielson

Department of Informatics and Mathematical Modelling
Danmarks Tekniske Universitet
Lyngby, Denmark
{cdpu, riis, nielson}@imm.dtu.dk

Abstract We present a systematic technique for transforming XACML 3.0 policies in Answer Set Programming (ASP). We show that the resulting logic program has a unique answer set that directly corresponds to our formalisation of the standard semantics of XACML 3.0 from [9]. We demonstrate how our results make it possible to use off-the-shelf ASP solvers to formally verify properties of access control policies represented in XACML, such as checking the completeness of a set of access control policies and verifying policy properties.

Keywords: XACML, access control, policy language, Answer Set Programming

1 Background

XACML (eXtensible Access Control Markup Language) is a prominent access control language that is widely adopted both in industry and academia. XACML is an international standard in the field of information security and in February 2005, XACML version 3.0 was ratified by OASIS.¹ XACML represents a shift from a more static security approach as exemplified by ACLs (Access Control Lists) towards a dynamic approach, based on Attribute Based Access Control (ABAC) systems. These dynamic security concepts are more difficult to understand, audit and interpret in real-world implications. The use of XACML requires not only the right tools but also well-founded concepts for policy creation and management.

The problem with XACML is that its specification is described in natural language (c.f. [11]) and manual analysis of the overall effect and consequences of a large XACML policy set is a very daunting and time-consuming task. How can a policy developer be certain that the represented policies capture all possible requests? Can they lead to conflicting decisions for some request? Do the policies satisfy all required properties? These complex problems cannot be solved easily without some automatised support.

To address this problem we propose a logic-based XACML analysis framework using Answer Set Programming (ASP). With ASP we model an XACML Policy Decision Point (PDP) that loads XACML policies and evaluates XACML requests against these policies. The expressivity of ASP and the existence of efficient implementations of the answer set semantics, such as `clasp`² and `DLV`³, provide the means for declarative specification and verification of properties of XACML policies.

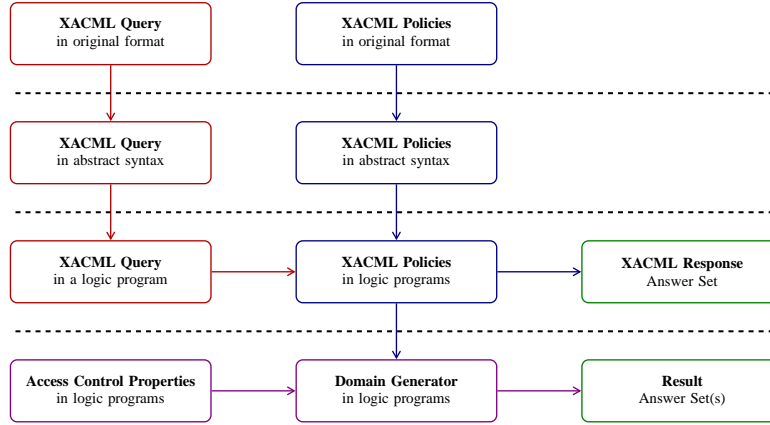
¹ The Organization for the Advancement of Structured Information Standards (OASIS) is a global consortium that drives the development, convergence, and adoption of e-business and web service standards.

² <http://www.cs.uni-potsdam.de/clasp/>

³ <http://www.dlvsystem.com/>

Our work is depicted in Figure 1. There are two main modules, viz. the PDP simulation module and the access control (AC) security property verification module. In the first module, we transform an XACML query and XACML policies from the original format in XML syntax into abstract syntax which is more compact than the original. Subsequently we generate a query program Π_Q and XACML policies program Π_{XACML} that correspond to the XACML query and the XACML policies, respectively. We show that the corresponding answer set (AS) of $\Pi_Q \cup \Pi_{XACML}$ is unique and it coincides with the semantics of original XACML policy evaluation. In the second module, we demonstrate how our results make it possible to use off-the-shelf ASP solvers to formally verify properties of AC policies represented in XACML. First we encode the AC security property and a generator for each possible domain of XACML policies into logic programs $\Pi_{AC-property}$ and $\Pi_{generator}$, respectively. The encoding of AC property is in the negated formula in order to show at a later stage that each answer set corresponds to a counter example that violates the AC property. Together with the combination of $\Pi_{XACML} \cup \Pi_{AC-property} \cup \Pi_{generator}$ we show that the XACML policies satisfy the AC property when there is no available answer set.

Figure 1. Translation Process from Original XACML to XACML-ASP



Outline. We consider the current version, XACML 3.0, Committee Specification 01, 10 August 2010. In Section 2 we explain the abstract syntax and semantics of XACML 3.0. Then we describe the transformation of XACML 3.0 components into logic programs in Section 3. We show the relation between XACML 3.0 semantics and the answer sets in Section 4. Next, in Section 5, we show how to verify AC properties, such as checking the completeness of a set of policies. In Section 6 we discuss the related work. We end the paper with conclusions and future work.

2 XACML 3.0

In order to avoid superfluous syntax of XACML 3.0, first we present the abstract syntax of XACML 3.0 which only shows the important components of XACML 3.0. We continue the explanation by presenting the semantics of XACML 3.0 components' evaluation based on Committee Specification [11]. We take the work of Ramli *et. al* work [9] as our reference.

2.1 Abstract Syntax of XACML 3.0

Table 1 shows the abstract syntax of XACML 3.0. We use bold font for non-terminal **symbols**, typewriter font for terminal symbols and *identifiers* and *values* are written in italic font. A symbol followed by the star symbol (*) indicates that there are zero or more occurrences of that symbol. Similarly, a symbol followed by the plus symbol (+) indicates that there are one or more occurrences of that symbol. We consider that each policy has a unique identifier (ID). We use initial capital letter for XACML components such as PolicySet, Policy, Rule, etc., and small letters for English terminology.

Table 1. Abstraction of XACML 3.0 Components

XACML Policy Components		
PolicySet	\mathcal{PS}	$::= \mathcal{PS}_{id} = [\mathcal{T}, \langle \mathcal{PS}_{id} \mid \mathcal{P}_{id} \rangle^*, \mathbf{CombID}]$
Policy	\mathcal{P}	$::= \mathcal{P}_{id} = [\mathcal{T}, \langle \mathcal{R}_{id} \rangle^+, \mathbf{CombID}]$
Rule	\mathcal{R}	$::= \mathcal{R}_{id} = [\mathbf{Effect}, \mathcal{T}, \mathcal{C}]$
Condition	\mathcal{C}	$::= \text{true} \mid f^{\text{bool}}(a_1, \dots, a_n)$
Target	\mathcal{T}	$::= \text{null} \mid \bigwedge \mathcal{E}^+$
AnyOf	\mathcal{E}	$::= \bigvee \mathcal{A}^+$
AllOf	\mathcal{A}	$::= \bigwedge \mathcal{M}^+$
Match	\mathcal{M}	$::= \mathcal{Attr}$
Attribute	\mathbf{CombID}	$::= \text{po} \mid \text{do} \mid \text{fa} \mid \text{ooa}$
	\mathbf{Effect}	$::= \text{p} \mid \text{d}$
	\mathcal{Attr}	$::= \text{category}(\text{attribute_value})$
XACML Request Component		
Request	\mathcal{Q}	$::= (\mathcal{Attr} \mid \text{error}(\mathcal{Attr}))^+$

There are three levels of policies in XACML, namely PolicySet, Policy and Rule. PolicySet or Policy can act as the root of a set of access control policies, while Rule is a single entity that describes one particular access control policy. Throughout this paper we consider that PolicySet is the root of the set of access control policies.

Both PolicySet and Policy function as containers for a sequence of PolicySet, Policy or Rule. A PolicySet contains either a sequence of PolicySet elements or a sequence of Policy elements, while a Policy can only contain a sequence of Rule elements. Every sequence of PolicySet, Policy or Rule elements has an associated *combining algorithm*. There are four common combining algorithms defined in XACML 3.0, namely *permit-overrides* (po), *deny-overrides* (do), *first-applicable* (fa) and *only-one-applicable* (ooa).

A Rule describes an individual access control policy. It regulates whether an access should be *permitted* (p) or *denied* (d). All PolicySet, Policy and Rule are applicable whenever their Target matches with the Request. When the Rule's Target matches the Request, then the applicability of the Rule is refined by its Condition.

A Target element identifies the set of decision requests that the parent element is intended to evaluate. The Target element must appear as a child of a PolicySet and Policy element and may appear as a child of a Rule element. The empty Target for Rule element is indicated by `null` attribute. The Target element contains a conjunctive sequence of AnyOf elements. The AnyOf element contains a disjunctive sequence of AllOf elements, while the AllOf element contains a conjunctive sequence of Match elements. Each Match element specifies an attribute that a Request should match.

A Condition is a Boolean function over attributes or functions of attributes. In this abstraction, the user is free to define the Condition as long as its expression returns a Boolean value, i.e., either true or false. Empty Condition is always associated to true.

A Request contains a set of attribute values for a particular access request and the error messages that occurred during the evaluation of attribute values.

2.2 XACML 3.0 Formal Semantics

The evaluation of XACML policies starts from the evaluation of Match elements and continues bottom-up until the evaluation of the root of the XACML element, i.e., the evaluation of PolicySet. For each XACML element X we denote by $\llbracket X \rrbracket$ a semantic function associated to X . To each Request element, this function assigns a value from a set of values that depends on the particular type of the XACML element X . For example, the semantic function $\llbracket X \rrbracket$, where X is a Match element, ranges over the set $\{m, nm, idt\}$, while its range is the set $\{t, f, idt\}$ when X is a Condition element. A further explanation will be given below. An XACML component returns an indeterminate value whenever the decision cannot be made. This happens when there is an error during the evaluation process. See [9] for further explanation of the semantics of XACML 3.0.

Evaluation of Match, Allof, AnyOf and Target Components. Let X be either a Match, an Allof, an AnyOf or a Target component and let \mathcal{Q} be a set of all possible Requests. A *Match semantic function* is a mapping $\llbracket X \rrbracket : \mathcal{Q} \rightarrow \{m, nm, idt\}$, where m , nm and idt denote *match*, *no-match* and *indeterminate*, respectively.

Our evaluation of Match element is based on equality function.⁴ We check whether there are any attribute values in Request element that match the Match attribute value.

Let \mathcal{Q} be a Request element and let \mathcal{M} be a Match element. The evaluation of Match \mathcal{M} is as follows

$$\llbracket \mathcal{M} \rrbracket(\mathcal{Q}) = \begin{cases} m & \text{if } \mathcal{M} \in \mathcal{Q} \text{ and } \text{error}(\mathcal{M}) \notin \mathcal{Q} \\ nm & \text{if } \mathcal{M} \notin \mathcal{Q} \text{ and } \text{error}(\mathcal{M}) \notin \mathcal{Q} \\ idt & \text{if } \text{error}(\mathcal{M}) \in \mathcal{Q} \end{cases} \quad (1)$$

The evaluation of Allof is a conjunction of a sequence of Match elements. The value of m , nm and idt corresponds to true, false and undefined in 3-valued logic, respectively.

Given a Request \mathcal{Q} , the evaluation of Allof, $\mathcal{A} = \bigwedge_{i=1}^n \mathcal{M}_i$, is as follows

$$\llbracket \mathcal{A} \rrbracket(\mathcal{Q}) = \begin{cases} m & \text{if } \forall i : \llbracket \mathcal{M}_i \rrbracket(\mathcal{Q}) = m \\ nm & \text{if } \exists i : \llbracket \mathcal{M}_i \rrbracket(\mathcal{Q}) = nm \\ idt & \text{otherwise} \end{cases} \quad (2)$$

where each \mathcal{M}_i is a Match element.

The evaluation of AnyOf element is a disjunction of a sequence of Allof elements. Given a Request \mathcal{Q} , the evaluation of AnyOf, $\mathcal{E} = \bigvee_{i=1}^n \mathcal{A}_i$, is as follows

$$\llbracket \mathcal{E} \rrbracket(\mathcal{Q}) = \begin{cases} m & \text{if } \exists i : \llbracket \mathcal{A}_i \rrbracket(\mathcal{Q}) = m \\ nm & \text{if } \forall i : \llbracket \mathcal{A}_i \rrbracket(\mathcal{Q}) = nm \\ idt & \text{otherwise} \end{cases} \quad (3)$$

where each \mathcal{A}_i is an Allof element.

⁴ Our Match evaluation is a simplification compared with [11].

The evaluation of Target element is a conjunction of a sequence of AnyOf elements. An empty Target, indicated by `null` attribute, is always evaluated to `m`. Given a Request \mathcal{Q} , the evaluation of Target, $\mathcal{T} = \bigwedge_{i=1}^n \mathcal{E}_i$, is as follows

$$\llbracket \mathcal{T} \rrbracket(\mathcal{Q}) = \begin{cases} m & \text{if } \forall i : \llbracket \mathcal{E}_i \rrbracket(\mathcal{Q}) = m \text{ or } \mathcal{T} = \text{null} \\ nm & \text{if } \exists i : \llbracket \mathcal{E}_i \rrbracket(\mathcal{Q}) = nm \\ \text{idt} & \text{otherwise} \end{cases} \quad (4)$$

where each \mathcal{E}_i is an AnyOf element.

Evaluation of Condition. Let X be a Condition component and let \mathbf{Q} be a set of all possible Requests. A *Condition semantic function* is a mapping $\llbracket X \rrbracket : \mathbf{Q} \rightarrow \{t, f, \text{idt}\}$, where `t`, `f` and `idt` denote *true*, *false* and *indeterminate*, respectively.

The evaluation of Condition element is based on the evaluation of its Boolean function as described in its element. To keep it abstract, we do not specify specific functions; however, we use an unspecified function, `eval`, that returns $\{t, f, \text{idt}\}$.

Given a Request \mathcal{Q} , the evaluation of Condition \mathcal{C} is as follows

$$\llbracket \mathcal{C} \rrbracket(\mathcal{Q}) = \text{eval}(\mathcal{C}, \mathcal{Q}) \quad (5)$$

Evaluation of Rule. Let X be a Rule component and let \mathbf{Q} be a set of possible Requests. A *Rule semantic function* is a mapping $\llbracket X \rrbracket : \mathcal{Q} \rightarrow \{p, d, i_p, i_d, na\}$, where `p`, `d`, `ip`, `id` and `na` correspond to *permit*, *deny*, *indeterminate permit*, *indeterminate deny* and *not – applicable*, respectively.

Given a Request \mathcal{Q} , the evaluation of Rule $\mathcal{R}_{id} = [E, \mathcal{T}, \mathcal{C}]$ is as follows

$$\llbracket \mathcal{R}_{id} \rrbracket(\mathcal{Q}) = \begin{cases} E & \text{if } \llbracket \mathcal{T} \rrbracket(\mathcal{Q}) = m \text{ and } \llbracket \mathcal{C} \rrbracket(\mathcal{Q}) = t \\ na & \text{if } (\llbracket \mathcal{T} \rrbracket(\mathcal{Q}) = m \text{ and } \llbracket \mathcal{C} \rrbracket(\mathcal{Q}) = f) \text{ or } \llbracket \mathcal{T} \rrbracket(\mathcal{Q}) = nm \\ i_E & \text{otherwise} \end{cases} \quad (6)$$

where E is an effect, $E \in \{p, d\}$, \mathcal{T} is a Target element and \mathcal{C} is a Condition element.

Evaluation of Policy and PolicySet. Let X be either a Policy or a PolicySet component and let \mathbf{Q} be a set of all possible Requests. A *Policy semantic function* is a mapping $\llbracket X \rrbracket : \mathcal{Q} \rightarrow \{p, d, i_p, i_d, i_{dp}, na\}$, where `p`, `d`, `ip`, `id`, `idp` and `na` correspond to *permit*, *deny*, *indeterminate permit*, *indeterminate deny*, *indeterminate deny permit* and *not – applicable*, respectively.

Given a Request \mathcal{Q} , the evaluation of Policy $\mathcal{P}_{id} = [T, \langle \mathcal{R}_1, \dots, \mathcal{R}_n \rangle, \text{CombID}]$ is as follows

$$\llbracket \mathcal{P}_{id} \rrbracket(\mathcal{Q}) = \begin{cases} i_d & \text{if } \llbracket \mathcal{T} \rrbracket(\mathcal{Q}) = \text{idt} \text{ and } \bigoplus_{\text{CombID}}(\mathbf{R}) = d \\ i_p & \text{if } \llbracket \mathcal{T} \rrbracket(\mathcal{Q}) = \text{idt} \text{ and } \bigoplus_{\text{CombID}}(\mathbf{R}) = p \\ na & \text{if } \llbracket \mathcal{T} \rrbracket(\mathcal{Q}) = nm \text{ or } \forall i : \llbracket \mathcal{R}_i \rrbracket(\mathcal{Q}) = na \\ \bigoplus_{\text{CombID}}(\mathbf{R}) & \text{otherwise} \end{cases} \quad (7)$$

where \mathcal{T} is a Target element, and each \mathcal{R}_i is a Rule element. We use \mathbf{R} to denote $\langle \llbracket \mathcal{R}_1 \rrbracket(\mathcal{Q}), \dots, \llbracket \mathcal{R}_n \rrbracket(\mathcal{Q}) \rangle$.

Note: The combining algorithm denoted by $\bigoplus_{\text{CombID}}$ will be explained in Sect. 2.3.

The evaluation of PolicySet is exactly like the evaluation of Policy except that it differs in terms of input parameter. While in Policy we use a sequence of Rule elements as an input, in the evaluation of PolicySet we use a sequence of Policy or PolicySet elements.

2.3 XACML Combining Algorithms

There are four common combining algorithms defined in XACML 3.0, namely permit-overrides (po), deny-overrides (do), first-applicable (fa) and only-one-applicable (ooa). In this paper, we do not consider the deny-overrides combining algorithm since it is the mirror of the permit-overrides combining algorithm.

Permit-Overrides (po) Combining Algorithm. The permit-overrides combining algorithm is intended for use if a permit decision should have priority over a deny decision. This algorithm has the following behaviour [11].

1. If any decision is “permit”, the result is “permit”.
2. Otherwise, if any decision is “indeterminate deny permit”, the result is “indeterminate deny permit”.
3. Otherwise, if any decision is “indeterminate permit” and another decision is “indeterminate deny” or “deny”, the result is “indeterminate deny permit”.
4. Otherwise, if any decision is “indeterminate permit”, the result is “indeterminate permit”.
5. Otherwise, if decision is “deny”, the result is “deny”.
6. Otherwise, if any decision is “indeterminate deny”, the result is “indeterminate deny”.
7. Otherwise, the result is “not applicable”.

Let $\langle s_1, \dots, s_n \rangle$ be a sequence of element of $\{p, d, i_p, i_d, i_{dp}, na\}$. The *permit-overrides combining operator* is defined as follows

$$\bigoplus_{po}(\langle s_1, \dots, s_n \rangle) = \begin{cases} p & \text{if } \exists i : s_i = p \\ i_{dp} & \text{if } \forall i : s_i \neq p \text{ and } (\exists j : s_j = i_{dp} \text{ or } (\exists j, j' : s_j = i_p \text{ and } (s_{j'} = i_d \text{ or } s_{j'} = d))) \\ i_p & \text{if } \exists i : s_i = i_p \text{ and } \forall j : s_j \neq i_p \Rightarrow s_j = na \\ d & \text{if } \exists i : s_i = d \text{ and } \forall j : s_j \neq d \Rightarrow (s_j = i_d \text{ or } s_j = na) \\ i_d & \text{if } \exists i : s_i = i_d \text{ and } \forall j : s_j \neq i_d \Rightarrow s_j = na \\ na & \text{otherwise} \end{cases} \quad (8)$$

First-Applicable (fa) Combining Algorithm. Each Rule must be evaluated in the order in which it is listed in the Policy. If a particular Rule is applicable, then the result of first-applicable combining algorithm must be the result of evaluating the Rule. If the Rule is “not applicable” then the next Rule in the order must be evaluated. If no further Rule in the order exists, then the first-applicable combining algorithm must return “not applicable”.

Let $\langle s_1, \dots, s_n \rangle$ be a sequence of element of $\{p, d, i_p, i_d, i_{dp}, na\}$. The *first-applicable combining operator* is defined as follows:

$$\bigoplus_{fa}(\langle s_1, \dots, s_n \rangle) = \begin{cases} s_i & \text{if } \exists i : s_i \neq na \text{ and } \forall j : (j < i) \Rightarrow (s_j = na) \\ na & \text{otherwise} \end{cases} \quad (9)$$

Only-One-Applicable (ooa) Combining Algorithm. If only one Policy is considered applicable by evaluation of its Target, then the result of the only-one-applicable combining algorithm must be the result of evaluating the Policy. If in the entire sequence of

Policy elements in the PolicySet, there is no Policy that is applicable, then the result of the only-one-applicable combining algorithm must be “not applicable”. If more than one Policy is considered applicable, then the result of the only-one-applicable combining algorithm must be “indeterminate”.

Let $\langle s_1, \dots, s_n \rangle$ be a sequence of element of $\{p, d, i_p, i_d, i_{dp}, na\}$. The *only-one-applicable combining operator* is defined as follows:

$$\bigoplus_{\text{ooa}}(\langle s_1, \dots, s_n \rangle) = \begin{cases} i_{dp} & \text{if } (\exists i : s_i = i_{dp}) \text{ or} \\ & (\exists i, j : i \neq j \text{ and } s_i = (d \text{ or } i_d) \wedge s_j = (p \text{ or } i_p)) \\ i_d & \text{if } (\forall i : s_i \neq (p \text{ or } i_p \text{ or } i_{dp})) \text{ and} \\ & ((\exists j : s_j = i_d) \text{ or } (\exists j, k : j \neq k \text{ and } s_j = s_k = d)) \\ i_p & \text{if } (\forall i : s_i \neq (d \text{ or } i_d \text{ or } i_{dp})) \text{ and} \\ & ((\exists j : s_j = i_p) \text{ or } (\exists j, k : j \neq k \text{ and } s_j = s_k = p)) \\ s_i & \text{if } \exists i : s_i \neq na \text{ and } \forall j : j \neq i \Rightarrow s_j = na \\ na & \text{otherwise} \end{cases} \quad (10)$$

3 Transforming XACML Components into Logic Programs

In this section we show, step by step, how to transform XACML 3.0 components into logic programs. We begin by introducing the syntax of logic programs (LPs). Then we show the transformation of XACML component into LPs starting from Request element to PolicySet element. We also present transformations for combining algorithms. The transformation of each XACML element is based on its formal semantics explained in Sect. 2.2 and Sect. 2.3.

3.1 Preliminaries

We recall basic notation and terminology that we use in the remainder of this paper.

First-Order Language. We consider an *alphabet* consisting of (finite or countably infinite) disjoint sets of variables, constants, function symbols, predicate symbols, connectives $\{\text{not}, \wedge, \leftarrow\}$, punctuation symbols $\{“(", “,”, “)”, “.”\}$ and special symbols $\{\top, \perp\}$. We use upper case letters to denote variables and lower case letters to denote constants, function and predicate symbols. Terms, atoms, literals and formulae are defined as usual. The *language* given by an alphabet consists of the set of all formulae constructed from the symbols occurring in the alphabet.

Logic Programs. A *rule* is an expression of the form

$$A \leftarrow B_1 \wedge \dots \wedge B_m \wedge \text{not } B_{m+1} \wedge \dots \wedge \text{not } B_n. \quad (11)$$

where A is either an atom or \perp and each B_i , $1 \leq i \leq n$, is an atom or \top . \top is a valid formula. We usually write $B_1 \wedge \dots \wedge B_m \wedge \text{not } B_{m+1} \wedge \dots \wedge \text{not } B_n$ simply as $B_1, \dots, B_m, \text{not } B_{m+1}, \dots, \text{not } B_n$. We call the rule as a *constraint* when $A = \perp$. One should observe that the body of a rule must not be empty. A *fact* is a rule of the form $A \leftarrow \top$.

A *logic program* is a finite set of rules. We denote $\text{ground}(II)$ for the set of all ground instances of rules in the program II .

3.2 XACML Components Transformation into Logic Programs

The transformation of XACML components is based on the semantics of each component explained in Sect. 2.2.

3.2.1 Request Transformation. *XACML Syntax:* Let $Q = \{ cat_1(a_1), \dots, cat_n(a_n) \}$ be a Request component. We transform all members of Request element into facts. The transformation of Request, Q , into LP Π_Q is as follows

$$cat_i(a_i) \leftarrow \top. \quad 1 \leq i \leq n$$

3.2.2 XACML Policy Components Transformation. We use a two-place function val to indicate the semantics of XACML components where the first argument is the name of XACML component and the second argument is its value. Please note that the calligraphic font in each transformation indicates the XACML component's name, that is, it does not represent a variable in LP.

Transformation of Match, AnyOf, AllOf and Target Components. Given a semantic equation of the form $\llbracket X \rrbracket_V(Q) = v$ if $cond_1$ and \dots and $cond_n$, we produce a rule of the form $val(X, v) \leftarrow cond_1, \dots, cond_n$. Given a semantic equation of the form $\llbracket X \rrbracket_V(Q) = v$ if $cond_1$ or \dots or $cond_n$, we produce a rule of the form $val(X, v) \leftarrow cond_i. \quad 1 \leq i \leq n$. For example, the Match evaluation $\llbracket \mathcal{M} \rrbracket(Q) = m$ if $cat(a) \in Q$ and $error(cat(a)) \notin Q$ is transformed into a rule in the form $val(\mathcal{M}, m) \leftarrow \mathcal{M}, \text{not } error(\mathcal{M})$. The truth value of \mathcal{M} depends on whether $\mathcal{M} \leftarrow \top$ is in Π_Q and the same is the case also for the truth value of $error(\mathcal{M})$.

Let \mathcal{M} be a Match component. The transformation of Match \mathcal{M} into LP $\Pi_{\mathcal{M}}$ is as follows (see (1) for Match evaluation)

$$\begin{aligned} val(\mathcal{M}, m) &\leftarrow \mathcal{M}, \text{not } error(\mathcal{M}). \\ val(\mathcal{M}, nm) &\leftarrow \text{not } cat(a), \text{not } error(\mathcal{M}). \\ val(\mathcal{M}, idt) &\leftarrow error(\mathcal{M}). \end{aligned}$$

Let $\mathcal{A} = \bigwedge_{i=1}^n \mathcal{M}_i$ be an AllOf component where each \mathcal{M}_i is a Match component. The transformation of AllOf \mathcal{A} into LP $\Pi_{\mathcal{A}}$ is as follows (see (2) for AllOf evaluation)

$$\begin{aligned} val(\mathcal{A}, m) &\leftarrow val(\mathcal{M}_1, m), \dots, val(\mathcal{M}_n, m). \\ val(\mathcal{A}, nm) &\leftarrow val(\mathcal{M}_i, nm). \quad (1 \leq i \leq n) \\ val(\mathcal{A}, idt) &\leftarrow \text{not } val(\mathcal{A}, m), \text{not } val(\mathcal{A}, nm). \end{aligned}$$

Let $\mathcal{E} = \bigvee_{i=1}^n \mathcal{A}_i$ be an AnyOf component where each \mathcal{A}_i is an AllOf component. The transformation of AnyOf \mathcal{E} into LP $\Pi_{\mathcal{E}}$ is as follows (see (3) for AnyOf evaluation)

$$\begin{aligned} val(\mathcal{E}, m) &\leftarrow val(\mathcal{A}_i, m). \quad (1 \leq i \leq n) \\ val(\mathcal{E}, nm) &\leftarrow val(\mathcal{A}_1, nm), \dots, val(\mathcal{A}_n, nm). \\ val(\mathcal{E}, idt) &\leftarrow \text{not } val(\mathcal{A}, m), \text{not } val(\mathcal{E}, nm). \end{aligned}$$

Let $\mathcal{T} = \bigwedge_{i=1}^n \mathcal{T}_i$ be a Target component where each \mathcal{E}_i is an AnyOf component. The transformation of Target \mathcal{T} into LP $\Pi_{\mathcal{T}}$ is as follows (see (4) for Target evaluation)

$$\begin{aligned} val(\text{null}, m) &\leftarrow \top. \\ val(\mathcal{T}, m) &\leftarrow val(\mathcal{E}_1, m), \dots, val(\mathcal{E}_n, m). \\ val(\mathcal{T}, nm) &\leftarrow val(\mathcal{E}_i, nm). \quad (1 \leq i \leq n) \\ val(\mathcal{T}, idt) &\leftarrow \text{not } val(\mathcal{T}, m), \text{not } val(\mathcal{T}, nm). \end{aligned}$$

Transformation of Condition Component. The transformation of Condition \mathcal{C} into LP $\Pi_{\mathcal{C}}$ is as follows

$$\text{val}(\mathcal{C}, V) \leftarrow \text{eval}(\mathcal{C}, V).$$

Moreover, the transformation of Condition also depends on the transformation of eval function into LP. Since we do not describe specific eval functions, we leave this transformation to the user.

Example 1. A possible eval function for "rule r1: patient only can see his or her patient record" is

$$\begin{aligned} \Pi_{\text{cond}(r1)} : \\ \text{val}(\text{cond}(r1), V) &\leftarrow \text{eval}(\text{cond}(r1), V). \\ \text{eval}(\text{cond}(r1), t) &\leftarrow \text{patient_id}(X), \text{patient_record_id}(X), \\ &\quad \text{not error}(\text{patient_id}(X)), \text{not error}(\text{patient_record_id}(X)). \\ \text{eval}(\text{cond}(r1), f) &\leftarrow \text{patient_id}(X), \text{patient_record_id}(Y), X \neq Y, \\ &\quad \text{not error}(\text{patient_id}(X)), \text{not error}(\text{patient_record_id}(Y)). \\ \text{eval}(\text{cond}(r1), \text{idt}) &\leftarrow \text{not eval}(\text{cond}(r1), t), \text{not eval}(\text{cond}(r1), f). \end{aligned}$$

The $\text{error}(\text{patient_id}(X))$ and $\text{error}(\text{patient_record_id}(X))$ indicate possible errors that might occur, e.g., the system could not connect to the database so that the system does not know the ID of the patient. \square

Transformation of Rule Component. The general step of the transformation of Rule component is similar to the transformation of Match component.

Let $\mathcal{R} = [e, \mathcal{T}, \mathcal{C}]$ be a Rule component where $e \in \{p, d\}$, \mathcal{T} is a Target and \mathcal{C} is a Condition. The transformation of Rule \mathcal{R} into LP $\Pi_{\mathcal{R}}$ is as follows (see (6) for Rule evaluation)

$$\begin{aligned} \text{val}(\mathcal{R}, e) &\leftarrow \text{val}(\mathcal{T}, m), \text{val}(\mathcal{C}, t). \\ \text{val}(\mathcal{R}, na) &\leftarrow \text{val}(\mathcal{T}, m), \text{val}(\mathcal{C}, f). \\ \text{val}(\mathcal{R}, na) &\leftarrow \text{val}(\mathcal{T}, nm). \\ \text{val}(\mathcal{R}, i_e) &\leftarrow \text{not val}(\mathcal{R}, e), \text{not val}(\mathcal{R}, na). \end{aligned}$$

Transformation of Policy and PolicySet Components. Given a Policy component $\mathcal{P}_{id} = [\mathcal{T}, \langle \mathcal{R}_1, \dots, \mathcal{R}_n \rangle, \text{CombID}]$ where \mathcal{T} is a Target, $\langle \mathcal{R}_1, \dots, \mathcal{R}_n \rangle$ is a sequence of Rule elements and CombID is a combining algorithm identifier. In order to indicate that the Policy contains Rule \mathcal{R}_i , for every Rule $\mathcal{R}_i \in \langle \mathcal{R}_1, \dots, \mathcal{R}_n \rangle$, $\Pi_{\mathcal{P}_{id}}$ contains:

$$\text{decision_of}(\mathcal{P}_{id}, \mathcal{R}_i, V) \leftarrow \text{val}(\mathcal{R}_i, V). (1 \leq i \leq n)$$

The transformation for Policy Π into LP $\Pi_{\mathcal{P}_{id}}$ is as follows (see (7) for Policy evaluation)

$$\begin{aligned} \text{val}(\mathcal{P}_{id}, i_d) &\leftarrow \text{val}(\mathcal{T}, \text{idt}), \text{algo}(\text{CombID}, \mathcal{P}_{id}, d). \\ \text{val}(\mathcal{P}_{id}, i_p) &\leftarrow \text{val}(\mathcal{T}, \text{idt}), \text{algo}(\text{CombID}, \mathcal{P}_{id}, p). \\ \text{val}(\mathcal{P}_{id}, na) &\leftarrow \text{val}(\mathcal{T}, nm). \\ \text{val}(\mathcal{P}_{id}, na) &\leftarrow \text{val}(\mathcal{R}_1, na), \dots, \text{val}(\mathcal{R}_n, na). \\ \text{val}(\mathcal{P}_{id}, V') &\leftarrow \text{val}(\mathcal{T}, m), \text{decision_of}(\mathcal{P}_{id}, \mathcal{R}, V), V \neq na, \text{algo}(\text{CombID}, \mathcal{P}_{id}, V'). \\ \text{val}(\mathcal{P}_{id}, V') &\leftarrow \text{val}(\mathcal{T}, \text{idt}), \text{decision_of}(\mathcal{P}_{id}, \mathcal{R}, V), V \neq na, \text{algo}(\text{CombID}, \mathcal{P}_{id}, V'), V' \neq p. \\ \text{val}(\mathcal{P}_{id}, V') &\leftarrow \text{val}(\mathcal{T}, \text{idt}), \text{decision_of}(\mathcal{P}_{id}, \mathcal{R}, V), V \neq na, \text{algo}(\text{CombID}, \mathcal{P}_{id}, V'), V' \neq d. \end{aligned}$$

We write a formula $\text{decision_of}(\mathcal{P}_{id}, \mathcal{R}, V), V \neq \text{na}$ to make sure that there is a Rule in the Policy that is not evaluated to na. We do this to avoid a return value from a combining algorithm that is not na, even though all of the Rule elements are evaluated to na. The transformation of PolicySet is similar to the transformation of Policy component.

3.3 Combining Algorithm Transformation

We define generic LPs for permit-overrides combining algorithm and only-one-applicable combining algorithm. Therefore, we use a variable P to indicate a variable over Policy identifier and R, R_1 and R_2 to indicate variables over Rule identifiers. In case the evaluation of PolicySet, the input P is for PolicySet identifier, R, R_1 and R_2 are for Policy (or PolicySet) identifiers.

Permit-Overrides Transformation. Let Π_{po} be a LP obtained by permit-overrides combining algorithm transformation (see (8) for the permit-overrides combining algorithm semantics). Π_{po} contains:

$$\begin{aligned} \text{algo}(\text{po}, P, p) &\leftarrow \text{decision_of}(P, R, p). \\ \text{algo}(\text{po}, P, i_{dp}) &\leftarrow \text{not algo}(\text{po}, P, p), \text{decision_of}(P, R, i_{dp}). \\ \text{algo}(\text{po}, P, i_{dp}) &\leftarrow \text{not algo}(\text{po}, P, p), \text{decision_of}(P, R_1, i_p), \text{decision_of}(P, R_2, d). \\ \text{algo}(\text{po}, P, i_{dp}) &\leftarrow \text{not algo}(\text{po}, P, p), \text{decision_of}(P, R_1, i_p), \text{decision_of}(P, R_2, i_d). \\ \text{algo}(\text{po}, P, i_p) &\leftarrow \text{not algo}(\text{po}, P, p), \text{not algo}(\text{po}, P, i_{dp}), \text{decision_of}(P, R, i_p). \\ \text{algo}(\text{po}, P, d) &\leftarrow \text{not algo}(\text{po}, P, p), \text{not algo}(\text{po}, P, i_{dp}), \text{not algo}(\text{po}, P, i_p), \\ &\quad \text{decision_of}(P, R, d). \\ \text{algo}(\text{po}, P, i_d) &\leftarrow \text{not algo}(\text{po}, P, p), \text{not algo}(\text{po}, P, i_{dp}), \text{not algo}(\text{po}, P, i_p), \\ &\quad \text{not algo}(\text{po}, P, d), \text{decision_of}(P, R, i_d). \\ \text{algo}(\text{po}, P, \text{na}) &\leftarrow \text{not algo}(\text{po}, P, p), \text{not algo}(\text{po}, P, i_{dp}), \text{not algo}(\text{po}, P, i_p), \\ &\quad \text{not algo}(\text{po}, P, d), \text{not algo}(\text{po}, P, i_d). \end{aligned}$$

First-Applicable Transformation. Let Π_{fa} be a logic program obtained by first-applicable combining algorithm transformation (see (9) for the first-applicable combining algorithm semantics). For each Policy (or PolicySet) which uses this combining algorithm, $\mathcal{P}_{id} = [\mathcal{T}, \langle \mathcal{R}_1, \dots, \mathcal{R}_n \rangle, \text{fa}]$, $\Pi_{\mathcal{P}_{id}}$ contains:

$$\begin{aligned} \text{algo}(\text{fa}, \mathcal{P}_{id}, E) &\leftarrow \text{decision_of}(\mathcal{P}_{id}, \mathcal{R}_1, V), V \neq \text{na}. \\ \text{algo}(\text{fa}, \mathcal{P}_{id}, E) &\leftarrow \text{decision_of}(\mathcal{P}_{id}, \mathcal{R}_1, \text{na}), \text{decision_of}(\mathcal{P}_{id}, \mathcal{R}_2, E), E \neq \text{na}. \\ &\vdots \\ \text{algo}(\text{fa}, \mathcal{P}_{id}, E) &\leftarrow \text{decision_of}(\mathcal{P}_{id}, \mathcal{R}_1, \text{na}), \dots, \text{decision_of}(\mathcal{P}_{id}, \mathcal{R}_{n-1}, \text{na}), \text{decision_of}(\mathcal{P}_{id}, \mathcal{R}_n, E). \end{aligned}$$

Only-One-Applicable Transformation. Let Π_{ooa} be a logic program obtained by only-one-applicable combining algorithm transformation (see (10) for the only-one-applicable

combining algorithm semantics). Π_{ooa} contains:

```

algo(ooa, P, idp) ← decision_of(P, R, idp).
algo(ooa, P, idp) ← decision_of(P, R1, id), decision_of(P, R2, ip), R1 ≠ R2.
algo(ooa, P, idp) ← decision_of(P, R1, id), decision_of(P, R2, p), R1 ≠ R2.
algo(ooa, P, idp) ← decision_of(P, R1, d), decision_of(P, R2, ip), R1 ≠ R2.
algo(ooa, P, idp) ← decision_of(P, R1, d), decision_of(P, R2, p), R1 ≠ R2.
algo(ooa, P, ip) ← not algo(ooa, P, idp), decision_of(P, R, ip).
algo(ooa, P, ip) ← not algo(ooa, P, idp), decision_of(P, R1, p), decision_of(P, R2, p), R1 ≠ R2.
algo(ooa, P, id) ← not algo(ooa, P, idp), decision_of(P, R, id).
algo(ooa, P, id) ← not algo(ooa, P, idp), decision_of(P, R1, d), decision_of(P, R2, d), R1 ≠ R2.
algo(ooa, P, p) ← not algo(ooa, P, idp), not (ooa, P, id), not (ooa, P, ip), decision_of(P, R, p).
algo(ooa, P, d) ← not algo(ooa, P, idp), not (ooa, P, id), not (ooa, P, ip), decision_of(P, R, d).
algo(ooa, P, na) ← not algo(ooa, P, idp), not (ooa, P, id), not (ooa, P, ip),
                    not decision_of(P, R, d), not decision_of(P, R, p).

```

4 Relation between XACML-ASP and XACML 3.0 Semantics

In this section we discuss the relationship between the ASP semantics and XACML 3.0 semantics. First, we recall the semantics of logic programs based on their answer sets. Then, we show that the program obtained from transforming XACML components into LPs (Π_{XACML}) merges with the query program (Π_Q) and has a unique answer set that the answer set corresponds to the semantics of XACML 3.0.

4.1 ASP Semantics

The declarative semantics of a logic program is given by a model-theoretic semantics of formulae in the underlying language. The formal definition of answer set semantics can be found in much literature such as [3,6].

The answer set semantics of logic program Π assigns to Π a collection of *answer sets* – interpretations of $\text{ground}(\Pi)$. An interpretation I of $\text{ground}(\Pi)$ is an answer set for Π if I is minimal (w.r.t. set inclusion) among the interpretations satisfying the rules of

$$\Pi^I = \{A \leftarrow B_1, \dots, B_m \mid A \leftarrow B_1, \dots, B_m, \text{not } B_{m+1}, \dots, \text{not } B_n \in \Pi \text{ and } I(\text{not } B_{m+1}, \dots, \text{not } B_n) = \text{true}\}$$

A logic program can have a single unique answer set, many or no answer set(s). Therefore, we show that programs with a particular characteristic are guaranteed to have a unique answer set.

Acyclic Programs. We say that a program is *acyclic* when there is no cycle in the program. The acyclicity in the program is guaranteed by the existence of a certain fixed assignment of natural numbers to atoms that is called a *level mapping*.

A *level mapping* for a program Π is a function

$$l : \mathcal{B}_\Pi \rightarrow \mathbb{N}$$

where \mathbb{N} is the set of natural numbers and \mathcal{B}_Π is the Herbrand base for Π . We extend the definition of level mapping to a mapping from ground literals to natural numbers by setting $l(\text{not } A) = l(A)$.

Let Π be a logic program and l be a level mapping for Π . Π is *acyclic with respect to l* if for every clause $A \leftarrow B_1, \dots, B_m, \text{not } B_{m+1}, \dots, \text{not } B_n$ in $\text{ground}(\Pi)$ we find

$$l(A) > l(B_i) \text{ for all } i \text{ with } 1 \leq i \leq n$$

Π is *acyclic* if it is acyclic with respect to some degree of level mapping. Acyclic programs are guaranteed to have a unique answer set [3].

4.2 XACML Semantics Based On ASP Semantics

We can see from Sect. 3 that all of the XACML 3.0 transformation programs are acyclic. Thus, it is guaranteed that Π_{XACML} has a unique answer set.

Proposition 1. *Let Π_{XACML} be a program obtained from XACML 3.0 element transformations and let Π_Q be a program transformation of Request Q . Let I be the answer set of $\Pi_{\text{XACML}} \cup \Pi_Q$. Then the following equation holds*

$$\llbracket X \rrbracket(Q) = V \text{ iff } \text{val}(X, V) \in I$$

where X is an XACML component.

Note: We can see that there is no cycle in all of the program transformations. Thus, there is a guarantee that the answer set of $\Pi_{\text{XACML}} \cup \Pi_Q$ is unique. The transformation of each component into a logic program is based on exactly the definition of its XACML evaluation. The proof of this proposition can be seen in the extended version in [10].

5 Analysis XACML Policies Using Answer Set Programming

In this section we show how to use ASP for analysing access control security properties through Π_{XACML} . In most cases, ASP solver can solve combinatorial problems efficiently. There are several combinatorial problems in analysis access control policies, e.g., gap-free property and conflict-free property [14,5]. In this section we look at gap-free analysis since in XACML 3.0 conflicts never occur.⁵ We also present a mechanism for the verification of security properties against a set of access control policies.

5.1 Query Generator

In order to analyse access control property, sometimes we need to analyse all possible queries that might occur. We use *cardinality constraint* (see [15,16]) to generate all possible values restored in the database for each attribute. For example, we have the following generator:

$$\begin{array}{ll} \mathcal{P}_{\text{generator}} : & \\ (1) \ 1\{\text{subject}(X) : \text{subject_db}(X)\}1 & \leftarrow \top. \\ (2) \ 1\{\text{action}(X) : \text{action_db}(X)\}1 & \leftarrow \top. \\ (3) \ 1\{\text{resource}(X) : \text{resource_db}(X)\}1 & \leftarrow \top. \\ (4) \ 1\{\text{environment}(X) : \text{environment_db}(X)\}1 & \leftarrow \top. \end{array}$$

The first line of the encoding means that we only consider one and only one *subject* attribute value obtained from the subject database. The rest of the encoding means the same as the *subject* attribute.

⁵ A conflict decision never occurs when we strictly use the standard combining algorithm defined in XACML 3.0, since every combining algorithm always return one value.

5.2 Gap-Free Analysis

A set of policies is *gap-free* if there is no access request for which there is an absence of decision. XACML defines that there is one PolicySet as the root of a set of policies. Hence, we say that there is a gap whenever we can find a request that makes the semantics of the \mathcal{PS}_{root} is assigned to na. We force ASP solver to find the gap by the following encoding.

$$\begin{aligned} \Pi_{gap} : \\ gap &\leftarrow \text{val}(\mathcal{PS}_{root}, \text{na}). \\ \perp &\leftarrow \text{not } gap. \end{aligned}$$

In order to make sure that a set of policies is gap-free we should generate all possible requests and test whether at least one request is not captured by the set of policies. Thus, the answer sets of program $\mathcal{P} = \Pi_{XACML} \cup \Pi_{generator} \cup \Pi_{gap}$ are witnesses that the set of policies encoded in Π_{XACML} is incomplete. When there is no model that satisfies the program then we are sure that the set of policies captures all of possible cases.

5.3 Property Analysis

The problem of verifying a security property Φ on XACML policies is not only to show that the property Φ holds on Π_{XACML} but also that we want to see the witnesses whenever the property Φ does not hold in order to help the policy developer refine the policies. Thus, we can see this problem as finding models for $\Pi_{XACML} \cup \Pi_{generator} \cup \Pi_{\neg\Phi}$. The founded model is the witness that the XACML policies cannot satisfy the property Φ .

Example 2. Suppose we have a security property:

Φ : An anonymous person **cannot** read any patient records.

Thus, the negation of property Φ is as follows

$\neg\Phi$: An anonymous person **can** read any patient records.

We define that anonymous persons are those who are neither patients, nor guardians, nor doctors, nor nurses. We encode $\mathcal{P}_{\neg\Phi}$ as follows

$$\begin{aligned} (1) \text{ anonymous} &\leftarrow \text{not } \text{subject}(\text{patient}), \text{not } \text{subject}(\text{guardian}), \\ &\quad \text{not } \text{subject}(\text{doctor}), \text{not } \text{subject}(\text{nurse}). \\ (2) \perp &\leftarrow \text{not } \text{anonymous}. \\ (3) \text{ action}(\text{read}) &\leftarrow \top. \\ (4) \text{ resource}(\text{patient_record}) &\leftarrow \top. \\ (5) \perp &\leftarrow \text{not } \text{val}(\mathcal{PS}_{root}, \text{p}). \end{aligned}$$

We list all of the requirements (lines 1 – 4). We force the program to find an anonymous person (line 2). Later we force that the returned decision should be to permit (line 5). When the program $\Pi_{XACML} \cup \Pi_{generator} \cup \Pi_{\neg\Phi}$ returns models, we conclude that the property Φ does not hold and the returned models are the flaws in the policies. On the other hand, we conclude that the property Φ is satisfied if no model is found.

6 Related Work

There are some approaches to defining AC policies in LPs, such as Barker *et al.* in [4] use constraint logic program to define role-based access control, Jajodia *et al.* in [7] using FAM / CAM program – a logical language that uses a fixed set of predicates. However, their approaches are based on their own access control policy language whereas our approach is to define a well-known access control policy language, XACML.

Our approach is inspired by the work of Ahn *et al.* [1,2]. There are three main differences between our approach and the work of Ahn *et al.*

First, while they consider XACML version 2.0 [8], we address the newer version, XACML 3.0. The main difference between XACML 3.0 and XACML 2.0 is the treatment of indeterminate values. As a consequence, the combining algorithms in XACML 3.0 are more complex than the ones in XACML 2.0. XACML 2.0 only has a single indeterminate value while XACML 3.0 distinguishes between the following three types of indeterminate values:

- i. *Indeterminate permit* (i_p) – an indeterminate value arising from a policy which could have been evaluated to permit but not deny;
- ii. *Indeterminate deny* (i_d) – an indeterminate value arising from a policy which could have been evaluated to deny but not permit;
- iii. *Indeterminate deny permit* (i_{dp}) – an indeterminate value arising from a policy which could have been evaluated as both deny and permit.

Second, Ahn *et al.* produce a monolithic logic program that can be used for the analysis of XACML policies while we take a more modular approach by first modelling an XACML PDP as a logic program and then using this encoding within a larger program for property analysis. While Ahn, *et al.* only emphasize the indeterminate value in the combining algorithms, our concern is “indeterminate” value in all aspect of XACML components, i.e., in Match, AnyOf, AllOf, Target, Condition, Rule, Policy and PolicySet components. Hence, we show that our main concern is to simulate the PDP as in XACML model.

Finally, Ahn *et al.* translate the XACML specification directly into logic programming, so the ambiguities in the natural language specification of XACML are also reflected in their encodings. To avoid this, we base our encodings on our formalisation of XACML from [9].

7 Conclusion and Future Work

We have modelled the XACML Policy Decision Point in a declarative way using the ASP technique by transforming XACML 3.0 elements into logic programs. Our transformation of XACML 3.0 elements is directly based on XACML 3.0 semantics [11] and we have shown that the answer set of each program transformation is unique and that it agrees with the semantics of XACML 3.0. Moreover, we can help policy developers analyse their access control policies such as checking policies’ completeness and verifying policy properties by inspecting the answer set of $\Pi_{XACML} \cup \Pi_{generator} \cup \Pi_{configuration}$ – the program obtained by transforming XACML 3.0 elements into logic programs joined with a query generator program and a configuration program.

For future work, we can extend our work to handle role-based access control in XACML 3.0 [13] and to handle delegation in XACML 3.0 [12]. Also, we can extend our work for checking reachability of policies. A policy is reachable if we can find a

request such that this policy is applicable. Thus, by removing unreachable policies we will not change the behaviour of the whole set of policies.

References

1. G.-J. Ahn, H. Hu, J. Lee, and Y. Meng. Reasoning about XACML policy descriptions in answer set programming (preliminary report). In *NMR'10*, 2010.
2. G.-J. Ahn, H. Hu, J. Lee, and Y. Meng. Representing and reasoning about web access control policies. In *COMPSAC*. IEEE Computer Society, 2010.
3. C. Baral. *Knowledge Representation, Reasoning and Declarative Problem Solving*. Cambridge University Press, 2003.
4. S. Barker and P. J. Stuckey. Flexible access control policy specification with constraint logic programming. *TISSEC*, 6, 2003.
5. G. Bruns and M. Huth. Access-control via Belnap logic: Effective and efficient composition and analysis. In *21st IEEE Computer Security Foundations Symposium*, 2008.
6. M. Gelfond. Handbook of knowledge representation. In B. Porter F. van Harmelen, V. Lifschitz, editor, *Foundations of Artificial Intelligence*, volume 3, chapter Answer Sets, pages 285–316. Elsevier, 2007.
7. S. Jajodia, P. Samarati, V. S. Subrahmanian, and E. Bertino. A unified framework for enforcing multiple access control policies. In *In Proceedings of ACM SIGMOD International Conference on Management of Data*, 1997.
8. T. Moses. eXtensible Access Control Markup Language (XACML) version 2.0. Technical report, OASIS, http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf, August 2010.
9. C. D. P. K. Ramli, H. R. Nielson, and F. Nielson. The logic of XACML. In *FACS'11*, Lecture Notes in Computer Science, 2011.
10. C. D. P. K. Ramli, H. R. Nielson, and F. Nielson. Xacml 3.0 in answer set programming – extended version. Technical report, arXiv.org, February 2013.
11. E. Rissanen. eXtensible Access Control Markup Language (XACML) version 3.0 (committe specification 01). Technical report, OASIS, <http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-cs-01-en.pdf>, August 2010.
12. E. Rissanen. XACML v3.0 administration and delegation profile version 1.0 (committe specification 01). Technical report, OASIS, <http://docs.oasis-open.org/xacml/3.0/xacml-3.0-administration-v1-spec-cs-01-en.pdf>, August 2010.
13. E. Rissanen. XACML v3.0 core and hierarchical role based access control (rbac) profile version 1.0 (committe specification 01). Technical report, OASIS, <http://docs.oasis-open.org/xacml/3.0/xacml-3.0-rbac-v1-spec-cs-01-en.pdf>, August 2010.
14. P. Samarati and Sabrina de Capitani di Vimercati. Access control: Policies, models, and mechanisms. In *Foundations of Security Analysis and Design, Tutorial Lectures*, 2001.
15. P. Simons, I. Niemelä, and T. Soininen. Extending and implementing the stable model semantics. *Artificial Intelligence*, 138:181–234, 2002.
16. T. Syrjänen. *Lparse 1.0 User's Manual*.

A ASP Semantics

A.1 Interpretations and Models

The *Herbrand Universe* $\mathcal{U}_{\mathcal{L}}$ for a language \mathcal{L} is the set of all ground terms that can be formed from the constants and function symbols appearing in \mathcal{L} . The *Herbrand base* $\mathcal{B}_{\mathcal{L}}$ for a language \mathcal{L} is the set of all ground atoms that can be formed by using

predicate symbols from \mathcal{L} and ground terms from $\mathcal{U}_{\mathcal{L}}$ as arguments. By \mathcal{B}_{Π} we denote the Herbrand base for language underlying the program Π . When the context is clear, we are safe to omit Π .

An *interpretation* I of a program Π is a mapping from the Herbrand base \mathcal{B}_{Π} to the set of truth values: true and false ($\{\top, \perp\}$). All atoms belong to interpretation I are mapped to \top . All atoms which does not occur in I are mapped to \perp .

The truth value of arbitrary formulae under some interpretation can be determined from a truth table as usual (see Table 2).

Table 2. Truth Values for Formulae

ϕ	ψ	not ϕ	$\phi \wedge \psi$	$\phi \leftarrow \psi$
\top	\top	\perp	\top	\top
\top	\perp	\perp	\perp	\top
\perp	\top	\top	\perp	\perp
\perp	\perp	\top	\perp	\top

The logical value of ground formulae can be derived from Table 2 in the usual way. A formula ϕ is then *true under interpretation* I , denoted by $I(\phi) = \top$, if all its ground instances are true in I ; it is *false under interpretation* I , denoted by $I(\phi) = \perp$, if there is a ground instance of ϕ that is false in I .

Let I be an interpretation. I *satisfies* formula ϕ if $I(\phi) = \top$. For a program Π , we say I *satisfies* Π if I satisfies for every rule in Π . An interpretation I is a *model* of formula ϕ if I satisfies ϕ .

Let \mathcal{I} be a collection of interpretations. Then an interpretation I in \mathcal{I} is called *minimal* in \mathcal{I} if and only if there is no interpretation J in \mathcal{I} such that $J \subsetneq I$. An interpretation I is called *least* in \mathcal{I} if and only if $I \subseteq J$ for any interpretation J in \mathcal{I} . A model M of a program Π is called *minimal* (respectively *least*) if it is minimal (respectively least) among all models of Π .

A.2 Answer Set

An interpretation I of $\text{ground}(\Pi)$ is an answer set for Π if I is minimal (w.r.t. set inclusion) among the interpretations satisfying the rules of

$$\Pi^I = \{A \leftarrow B_1, \dots, B_m \mid A \leftarrow B_1, \dots, B_m, \text{not } B_{m+1}, \dots, \text{not } B_n \in \Pi \text{ and } I(\text{not } B_{m+1}, \dots, \text{not } B_n) = \top\}$$

B Proofs

Lemma 1. *Let M be an answer set of program Π and let $H \leftarrow \text{Body}$ be a rule in Π . Then, $H \in M$ if $M(\text{Body}) = \top$.*

Proof. Let $\text{Body} = B_1, \dots, B_m, \text{not } B_{m+1}, \dots, \text{not } B_n$. To show the lemma holds, suppose $M(\text{Body}) = \top$. Then we find that $\{B_1, \dots, B_m\} \subseteq M$ and $M \cap \{B_{m+1}, \dots, B_n\} =$

\emptyset . Since M is a minimal model of Π^M then we find that $H \leftarrow B_1, \dots, B_n$ is in Π^M . Since $\{B_1, \dots, B_n\} \subseteq M$ and M is a model then $M(H) = \top$. Thus $H \in M$. \square

The Lemma 1 only ensures that if the body of a rule is true under an answer set M then the head is also in M . However, in general, if the head of a rule is in a answer set M then there is no guarantee that the body is always true under M . For example, suppose we have a program $\{p \leftarrow \top, p \leftarrow q\}$. In this example the only answer set is $M = \{p\}$. We can see that p is in M . However, q is not in M , thus, $M(q)$ is false.

Lemma 2. *Let M be an answer set of program Π and let H be in M . Then, there is a rule in Π where H as the head.*

Proof. Suppose that M is an answer set of program Π . Then we find that M is a minimal model of Π^M . Suppose $H \in M$ and there is no rule in Π^M such that H as the head. Then, we find that $M' = M / \{H\}$ and M' is a model of Π^M . Since M is a minimal model of Π^M but we have $M' \subset M$. Therefore we find a contradiction. Thus, there should be a rule in Π^M such that H as the head. Hence, there is a rule in Π such that H as the head. \square

Lemma 3. *Let M be an answer set of program Π and let H be in M . Then, there exists a rule where H as the head and the body is true under M .*

Proof. Suppose that M is an answer set of program Π . Since H is in M thus, by Lemma 2, we find that there is a rule in Π in a form $H \leftarrow Body$. Suppose that $M(Body) \neq \top$. Therefore, $H \leftarrow Body$ is not in Π^M . Moreover, we can find another interpretation M' such that $M / \{H\}$ and M' is also a model of Π^M . However, we know that M is a minimal model for Π^M but we have $M' \subset M$. Thus, there is a contradiction. \square

We define some notation:

XACML Components	XACML Symbols	LP Symbols
Match	\mathcal{M}	$\Pi^{\mathcal{M}} = \Pi_{\mathcal{M}}$
AllOf	$\mathcal{A} = \bigwedge \mathcal{M}_i$	$\Pi^{\mathcal{A}} = \bigcup \Pi^{\mathcal{M}_i} \cup \Pi_{\mathcal{A}}$
AnyOf	$\mathcal{E} = \bigvee \mathcal{A}_i$	$\Pi^{\mathcal{E}} = \bigcup \Pi^{\mathcal{A}_i} \cup \Pi_{\mathcal{E}}$
Target	$\mathcal{T} = \bigwedge \mathcal{E}_i$	$\Pi^{\mathcal{T}} = \bigcup \Pi^{\mathcal{E}_i} \cup \Pi_{\mathcal{T}}$
Condition	\mathcal{C}	$\Pi^{\mathcal{C}} = \Pi_{\mathcal{C}}$
Rule	$\mathcal{R} = [E, \mathcal{T}, \mathcal{C}]$	$\Pi^{\mathcal{R}} = \Pi^{\mathcal{T}} \cup \Pi^{\mathcal{C}} \cup \Pi_{\mathcal{R}}$
Policy	$\mathcal{P} = [\mathcal{T}, \langle \mathcal{R}_1, \dots, \mathcal{R}_n \rangle, \text{CombID}]$	$\Pi^{\mathcal{P}} = \bigcup \Pi^{\mathcal{R}_i} \cup \Pi^{\mathcal{T}} \cup \Pi^{\text{CombID}} \cup \Pi_{\mathcal{P}}$
PolicySet	$\mathcal{PS} = [\mathcal{T}, \langle \mathcal{P}_1, \dots, \mathcal{P}_n \rangle, \text{CombID}]$	$\Pi^{\mathcal{P}} = \bigcup \Pi^{\mathcal{P}_i} \cup \Pi^{\mathcal{T}} \cup \Pi^{\text{CombID}} \cup \Pi_{\mathcal{PS}}$
Combining Algorithm	CombID is either po or faor ooa	$\Pi^{\text{CombID}} = \bigcup \Pi_{\mathcal{R}_i} \cup \Pi_{\mathcal{P}_j} \Pi_{\text{CombID}}$

Match Evaluation.

Lemma 4. *Let $\Pi = \Pi_Q \cup \Pi^{\mathcal{M}}$ be a program and M be an answer set of Π . Then,*

$$\llbracket \mathcal{M} \rrbracket(Q) = m \quad \text{if and only if} \quad \text{val}(\mathcal{M}, m) \in M.$$

Proof. (\Rightarrow) Suppose that $\llbracket \mathcal{M} \rrbracket = m$ holds. Then, as defined in (1), $\mathcal{M} \in Q$ and $\text{error}(\mathcal{M}) \notin Q$. Based on the transformation of Request element, we find out that $\mathcal{M} \leftarrow \top$ is in Π and there is no rule where $\text{error}(\mathcal{M})$ as the head in Π . Since M is the minimal model of Π , we get that $\mathcal{M} \in M$ and $\text{error}(\mathcal{M}) \notin M$. Thus we get that

$M(\mathcal{M} \wedge \text{not error}(\mathcal{M})) = \top$. Therefore, by Lemma, $1 \text{ val}(\mathcal{M}, m) \in M$.
 (\Leftarrow) Suppose that $\text{val}(\mathcal{M}, m) \in M$. By Lemma 3 we get that there is a rule where $\text{val}(\mathcal{M}, m)$ as the head and the body is true under M . Since there is only one rule where $\text{val}(\mathcal{M}, m)$ as the head in Π , i.e., $\text{val}(\mathcal{M}, m) \leftarrow \mathcal{M}, \text{not error}(\mathcal{M})$, then, we find that $M(\mathcal{M} \wedge \text{not error}(\mathcal{M})) = \top$. Therefore, $\mathcal{M} \in M$ and $\text{error}(\mathcal{M}) \notin M$. Since the only possible to have \mathcal{M} true in this case is only through the Request transformation, we get that $\mathcal{M} \in \mathcal{Q}$ and $\text{error}(\mathcal{M}) \notin \mathcal{Q}$. Therefore, we obtain $\llbracket \mathcal{M} \rrbracket(\mathcal{Q}) = m$. \square

Lemma 5. Let $\Pi = \Pi_{\mathcal{Q}} \cup \Pi^{\mathcal{M}}$ be a program and M be an answer set of Π . Then,

$$\llbracket \mathcal{M} \rrbracket(\mathcal{Q}) = nm \quad \text{if and only if} \quad \text{val}(\mathcal{M}, nm) \in M .$$

Proof. (\Rightarrow) Suppose that $\llbracket \mathcal{M} \rrbracket = nm$. Then, as defined in (1) we have that $\mathcal{M} \notin \mathcal{Q}$ and $\text{error}(\mathcal{M}) \notin \mathcal{Q}$. Based on the transformation of Request, we find out that there is no rule where \mathcal{M} and $\text{error}(\mathcal{M})$ as the heads. Since M is the minimal model of Π , we get that \mathcal{M} and $\text{error}(\mathcal{M})$ are not in M . Thus, we get that $M(\text{not } \mathcal{M} \wedge \text{not error}(\mathcal{M})) = \top$. Therefore, by Lemma 1, $\text{val}(\mathcal{M}, nm) \in M$.

(\Leftarrow) Suppose that $\text{val}(\mathcal{M}, nm) \in M$ where $\mathcal{M} = \mathcal{M}$. Based on Lemma 3 we get that there is a rule where $\text{val}(\mathcal{M}, nm)$ as the head and the body is true under M . Since there is only one rule where $\text{val}(\mathcal{M}, nm)$ as the head in Π , i.e., $\text{val}(\mathcal{M}, nm) \leftarrow \text{not } \mathcal{M}, \text{not error}(\mathcal{M})$, then, we find that $M(\text{not } \mathcal{M} \wedge \text{not error}(\mathcal{M})) = \top$. Therefore, $\mathcal{M} \notin M$ and $\text{error}(\mathcal{M}) \notin M$. Since the only possible of declaring facts in this case is only through the Request transformation, we get that $\mathcal{M} \notin \mathcal{Q}$ and $\text{error}(\mathcal{M}) \notin \mathcal{Q}$. Therefore, we obtain $\llbracket \mathcal{M} \rrbracket(\mathcal{Q}) = nm$. \square

Lemma 6. Let $\Pi = \Pi_{\mathcal{Q}} \cup \Pi^{\mathcal{M}}$ be a program and M be an answer set of Π . Then,

$$\llbracket \mathcal{M} \rrbracket(\mathcal{Q}) = \text{idt} \quad \text{if and only if} \quad \text{val}(\mathcal{M}, \text{idt}) \in M .$$

Proof. (\Rightarrow) Suppose that $\llbracket \mathcal{M} \rrbracket(\mathcal{Q}) = \text{idt}$ holds where $\mathcal{M} = \mathcal{M}$. Then, as defined in (1), we have that $\text{error}(\mathcal{M}) \in \mathcal{Q}$. Based on the transformation of Request element, we find out that $\text{error}(\mathcal{M}) \leftarrow \top$ is in Π . Since M is the minimal model of Π , then, we get that $\text{error}(\mathcal{M}) \in M$. Thus, we get that $M(\text{error}(\mathcal{M})) = \top$. Therefore, $\text{val}(\mathcal{M}, \text{idt}) \in M$ since M is the minimal model of Π .

(\Leftarrow) Suppose that $\text{val}(\mathcal{M}, \text{idt}) \in M$. Based on Lemma 3 we get that there is a rule where $\text{val}(\mathcal{M}, \text{idt})$ as the head and the body is true under M . Since there is only one rule in Π with $\text{val}(\mathcal{M}, \text{idt})$ in the head, i.e., $\text{val}(\mathcal{M}, \text{idt}) \leftarrow \text{error}(\mathcal{M})$, then we find that $M(\text{error}(\mathcal{M})) = \top$. Therefore, $\text{error}(\mathcal{M}) \in M$. Since the only possible to have $\text{error}(\mathcal{M})$ true in this case is only through the Request transformation, we get that $\text{error}(\mathcal{M}) \in \mathcal{Q}$. Therefore, we obtain $\llbracket \mathcal{M} \rrbracket(\mathcal{Q}) = \text{idt}$. \square

Proposition 2. Let $\Pi = \Pi_{\mathcal{Q}} \cup \Pi^{\mathcal{M}}$ be a program and M be an answer set of Π . Then,

$$\llbracket \mathcal{M} \rrbracket(\mathcal{Q}) = V \quad \text{if and only if} \quad \text{val}(\mathcal{M}, V) \in M .$$

Proof. It follows from Lemma 4, Lemma 5 and Lemma 6 since the value of V only has three possibilities, i.e., $\{ m, nm, \text{idt} \}$. \square

Allof Evaluation.

Lemma 7. Let $\Pi = \Pi_{\mathcal{Q}} \cup \Pi^{\mathcal{A}}$ be a program and M be an answer set of Π . Then,

$$\llbracket \mathcal{A} \rrbracket(\mathcal{Q}) = m \quad \text{if and only if} \quad \text{val}(\mathcal{A}, m) \in M .$$

Proof. Let $\mathcal{A} = \bigwedge_{i=1}^n \mathcal{M}_i$.

(\Rightarrow) Suppose that $\llbracket \mathcal{A} \rrbracket(\mathcal{Q}) = m$ holds. Then, as defined in (2), $\forall i : \llbracket \mathcal{M}_i \rrbracket(\mathcal{Q}) = m, 1 \leq i \leq n$. Based on Prop. 2, $\forall i : \text{val}(\mathcal{M}_i, m) \in M, 1 \leq i \leq n$. Therefore, $M(\text{val}(\mathcal{M}_1, m) \wedge \dots \wedge \text{val}(\mathcal{M}_n, m)) = \top$. Hence, by Lemma 1, $\text{val}(\mathcal{A}, m) \in M$.

(\Leftarrow) Suppose that $\text{val}(\mathcal{A}, m) \in M$. Based on Lemma 3, there is a rule where $\text{val}(\mathcal{A}, m)$ as the head and the body is true under M . Since there is only one rule in Π with $\text{val}(\mathcal{A}, m)$ in the head, i.e., $\text{val}(\mathcal{A}, m) \leftarrow \text{val}(\mathcal{M}_1, m), \dots, \text{val}(\mathcal{M}_n, m)$, we find that $M(\text{val}(\mathcal{M}_1, m) \wedge \dots \wedge \text{val}(\mathcal{M}_n, m)) = \top$. Therefore, $\text{val}(\mathcal{M}_i, m) \in M, 1 \leq i \leq n$. Based on Prop. 2, $\llbracket \mathcal{M}_i \rrbracket(\mathcal{Q}) = m, 1 \leq i \leq n$. Therefore, based on (2), we obtain $\llbracket \mathcal{A} \rrbracket(\mathcal{Q}) = m$. \square

Lemma 8. Let $\Pi = \Pi_{\mathcal{Q}} \cup \Pi^{\mathcal{A}}$ be a program and M be an answer set of Π . Then,

$$\llbracket \mathcal{A} \rrbracket(\mathcal{Q}) = nm \quad \text{if and only if} \quad \text{val}(\mathcal{A}, nm) \in M .$$

Proof. Let $\mathcal{A} = \bigwedge_{i=1}^n \mathcal{M}_i$.

(\Rightarrow) Suppose that $\llbracket \mathcal{A} \rrbracket(\mathcal{Q}) = nm$ holds. Then, as defined in (2) we have that $\exists i : \llbracket \mathcal{M}_i \rrbracket(\mathcal{Q}) = nm$. Based on Prop. 2 we get that $\exists i : \text{val}(\mathcal{M}_i, nm) \in M$. Thus, we get that $\exists i : M(\text{val}(\mathcal{M}_i), nm) = \top$. Therefore, by Lemma 1, $\text{val}(\mathcal{A}, nm) \in M$.

(\Leftarrow) Suppose that $\text{val}(\mathcal{A}, nm) \in M$. Based on Lemma 3 we get that there is a rule where $\text{val}(\mathcal{A}, nm)$ as the head and the body is true under M . Based on AllOf transformation, $\exists i : M(\text{val}(\mathcal{M}_i), nm) = \top$. Therefore, $\exists i : \text{val}(\mathcal{M}_i, nm) \in M$. Based on Prop. 2 we get that $\exists i : \llbracket \mathcal{M}_i \rrbracket(\mathcal{Q}) = nm$. Therefore, based on (2), we obtain $\llbracket \mathcal{A} \rrbracket(\mathcal{Q}) = nm$. \square

Lemma 9. Let $\Pi = \Pi_{\mathcal{Q}} \cup \Pi^{\mathcal{A}}$ be a program and M be an answer set of Π . Then,

$$\llbracket \mathcal{A} \rrbracket(\mathcal{Q}) = \text{idt} \quad \text{if and only if} \quad \text{val}(\mathcal{A}, \text{idt}) \in M .$$

Proof. (\Rightarrow) Suppose that $\llbracket \mathcal{A} \rrbracket(\mathcal{Q}) = \text{idt}$. Then, as defined in (2), $\llbracket \mathcal{A} \rrbracket(\mathcal{Q}) \neq m$ and $\llbracket \mathcal{A} \rrbracket(\mathcal{Q}) \neq nm$. Thus, by Lemma 7 and Lemma 8, $\text{val}(\mathcal{A}, m) \notin M$ and $\text{val}(\mathcal{A}, nm) \notin M$. Hence, $M(\text{not val}(\mathcal{A}, m) \wedge \text{not val}(\mathcal{A}, nm)) = \top$. Therefore, by Lemma 1, $\text{val}(\mathcal{A}, \text{idt}) \in M$.

(\Leftarrow) Suppose that $\text{val}(\mathcal{A}, \text{idt}) \in M$. Based on Lemma 3, there is a rule where $\text{val}(\mathcal{A}, \text{idt})$ as the head and the body is true under M . There is only one rule where $\text{val}(\mathcal{A}, \text{idt})$ as the head in Π , i.e., $\text{val}(\mathcal{A}, \text{idt}) \leftarrow \text{not val}(\mathcal{A}, m), \text{not val}(\mathcal{A}, nm)$. Hence, $\text{val}(\mathcal{A}, m) \notin M$ and $\text{val}(\mathcal{A}, nm) \notin M$. Based on Lemma 7 and Lemma 8 we get that $\llbracket \mathcal{A} \rrbracket(\mathcal{Q}) \neq m$ and $\llbracket \mathcal{A} \rrbracket(\mathcal{Q}) \neq nm$. Therefore, based on (2), we obtain $\llbracket \mathcal{A} \rrbracket(\mathcal{Q}) = \text{idt}$. \square

Proposition 3. Let $\Pi = \Pi_{\mathcal{Q}} \cup \Pi^{\mathcal{A}}$ be a program obtained by merging Request transformation program $\Pi_{\mathcal{Q}}$ and AllOf \mathcal{A} transformations program with all of its components $\Pi^{\mathcal{A}}$. Let M be an answer set of Π . Then,

$$\llbracket \mathcal{A} \rrbracket(\mathcal{Q}) = V \quad \text{if and only if} \quad \text{val}(\mathcal{A}, V) \in M .$$

Proof. It follows from Lemma 7, Lemma 8 and Lemma 9 since the value of V only has three possibilities, i.e., $\{m, nm, \text{idt}\}$. \square

AnyOf Evaluation.

Lemma 10. Let $\Pi = \Pi_{\mathcal{Q}} \cup \Pi^{\mathcal{E}}$ be a program and M be an answer set of Π . Then,

$$\llbracket \mathcal{E} \rrbracket(\mathcal{Q}) = m \quad \text{if and only if} \quad \text{val}(\mathcal{E}, m) \in M .$$

Proof. Let $\mathcal{E} = \bigvee_{i=1}^n \mathcal{A}_i$

(\Rightarrow) Suppose that $\llbracket \mathcal{E} \rrbracket(\mathcal{Q}) = m$ holds. Then, as defined in (3), $\exists i : \llbracket \mathcal{A}_i \rrbracket(\mathcal{Q}) = m, 1 \leq i \leq n$. Based on Prop. 3, $\exists i : \text{val}(\mathcal{A}_i, m) \in M, 1 \leq i \leq n$. Thus, $\exists i : M(\text{val}(\mathcal{A}_i, m)) = \top$. Therefore, by Lemma 1, $\text{val}(\mathcal{E}, m) \in M$.

(\Leftarrow) Suppose that $\text{val}(\mathcal{E}, m) \in M$. Based on Lemma 3, here is a rule where $\text{val}(\mathcal{E}, m)$ as the head and the body is true under M . Based on AnyOf transformation, $\exists i : M(\text{val}(\mathcal{E}_i), m) = \top$. Therefore, $\exists i : \text{val}(\mathcal{E}_i, m) \in M$. Based on Prop. 3, $\exists i : \llbracket \mathcal{E}_i \rrbracket(\mathcal{Q}) = m$. Therefore, based on (3), we obtain $\llbracket \mathcal{E} \rrbracket(\mathcal{Q}) = m$. \square

Lemma 11. Let $\Pi = \Pi_{\mathcal{Q}} \cup \Pi^{\mathcal{E}}$ be a program and M be an answer set of Π . Then,

$$\llbracket \mathcal{E} \rrbracket(\mathcal{Q}) = nm \quad \text{if and only if} \quad \text{val}(\mathcal{E}, nm) \in M .$$

Proof. Let $\mathcal{E} = \bigvee_{i=1}^n \mathcal{A}_i$

(\Rightarrow) Suppose that $\llbracket \mathcal{E} \rrbracket(\mathcal{Q}) = nm$ holds. Then, as defined in (3), $\forall i : \llbracket \mathcal{A}_i \rrbracket(\mathcal{Q}) = nm$. Based on Prop. 3, $\forall i : \text{val}(\mathcal{A}_i, nm) \in M$. Thus, $M(\text{val}(\mathcal{A}_1, nm) \wedge \dots \wedge \text{val}(\mathcal{A}_n, nm)) = \top$. Therefore, by Lemma 1, $\text{val}(\mathcal{E}, nm) \in M$.

(\Leftarrow) Suppose that $\text{val}(\mathcal{E}, nm) \in M$. By Lemma 3, there is a rule where $\text{val}(\mathcal{E}, nm)$ as the head and the body is true under M . There is only one rule in Π with $\text{val}(\mathcal{E}, m)$ in the head in Π , i.e., $\text{val}(\mathcal{E}, nm) \leftarrow \text{val}(\mathcal{A}_1, nm), \dots, \text{val}(\mathcal{A}_n, nm)$. Thus, $M(\text{val}(\mathcal{A}_1, nm) \wedge \dots \wedge \text{val}(\mathcal{A}_n, nm)) = \top$. Therefore, $\text{val}(\mathcal{A}_i, nm) \in M, 1 \leq i \leq n$. Based on Prop. 3, $\llbracket \mathcal{A}_i \rrbracket(\mathcal{Q}) = nm, 1 \leq i \leq n$. Therefore, based on (3), we obtain $\llbracket \mathcal{E} \rrbracket(\mathcal{Q}) = nm$. \square

Lemma 12. Let $\Pi = \Pi_{\mathcal{Q}} \cup \Pi^{\mathcal{E}}$ be a program and M be an answer set of Π . Then,

$$\llbracket \mathcal{E} \rrbracket(\mathcal{Q}) = \text{idt} \quad \text{if and only if} \quad \text{val}(\mathcal{E}, \text{idt}) \in M .$$

Proof. (\Rightarrow) Suppose that $\llbracket \mathcal{E} \rrbracket(\mathcal{Q}) = \text{idt}$. Then, as defined in (3), $\llbracket \mathcal{E} \rrbracket(\mathcal{Q}) \neq m$ and $\llbracket \mathcal{E} \rrbracket(\mathcal{Q}) \neq nm$. Thus, by Lemma 10 and Lemma 11, $\text{val}(\mathcal{E}, m) \notin M$ and $\text{val}(\mathcal{E}, nm) \notin M$. Hence, $M(\text{not val}(\mathcal{E}, m) \wedge \text{not val}(\mathcal{E}, nm)) = \top$. By Lemma 1, $\text{val}(\mathcal{E}, \text{idt}) \in M$. (\Leftarrow) Suppose that $\text{val}(\mathcal{E}, \text{idt}) \in M$. Based on Lemma 3, there is a rule where $\text{val}(\mathcal{E}, \text{idt})$ as the head and the body is true under M . There is only one rule in Π with $\text{val}(\mathcal{E}, \text{idt})$ in the head, i.e., $\text{val}(\mathcal{E}, \text{idt}) \leftarrow \text{not val}(\mathcal{E}, m), \text{not val}(\mathcal{E}, nm)$. Hence, $\text{val}(\mathcal{E}, m) \notin M$ and $\text{val}(\mathcal{E}, nm) \notin M$. Based on Lemma 10 and Lemma 11, $\llbracket \mathcal{E} \rrbracket(\mathcal{Q}) \neq m$ and $\llbracket \mathcal{E} \rrbracket(\mathcal{Q}) \neq nm$. Therefore, based on (3), we obtain $\llbracket \mathcal{E} \rrbracket(\mathcal{Q}) = \text{idt}$. \square

Proposition 4. Let $\Pi = \Pi_{\mathcal{Q}} \cup \Pi^{\mathcal{E}}$ be a program obtained by merging Request transformation program $\Pi_{\mathcal{Q}}$ and AnyOf \mathcal{E} transformations program with all of its components $\Pi^{\mathcal{E}}$. Let M be an answer set of Π . Then,

$$\llbracket \mathcal{E} \rrbracket(\mathcal{Q}) = V \quad \text{if and only if} \quad \text{val}(\mathcal{E}, V) \in M .$$

Proof. It follows from Lemma 10, Lemma 11 and Lemma 12 since the value of V only has three possibilities, i.e., $\{ m, nm, \text{idt} \}$. \square

Target Evaluation.

Lemma 13. Let $\Pi = \Pi_{\mathcal{Q}} \cup \Pi^{\mathcal{T}}$ be a program and M be an answer set of Π . Then,

$$\llbracket \mathcal{T} \rrbracket(\mathcal{Q}) = m \quad \text{if and only if} \quad \text{val}(\mathcal{T}, m) \in M .$$

Proof. Let $\mathcal{T} = \bigwedge_{i=1}^n \mathcal{E}_i$.

(\Rightarrow) Suppose that $\llbracket \mathcal{T} \rrbracket(\mathcal{Q}) = m$ holds. Then, as defined in (4), we have that

1. $\forall i : \llbracket \mathcal{E}_i \rrbracket(\mathcal{Q}) = m, 1 \leq i \leq n$. Based on Prop. 4, $\forall i : \text{val}(\mathcal{E}_i, m) \in M, 1 \leq i \leq n$. Thus, $M(\text{val}(\mathcal{E}_1, m) \wedge \dots \wedge \text{val}(\mathcal{E}_n, m)) = \top$. Therefore, by Lemma 1, $\text{val}(\mathcal{T}, m) \in M$.
2. $\mathcal{T} = \text{null}$. Based on Target transformation we get that $\text{val}(\text{null}, m) \leftarrow \top$. Thus, $\text{val}(\mathcal{T}, m) \in M$ since M is the minimal model of Π .

(\Leftarrow) Suppose that $\text{val}(\mathcal{T}, m) \in M$. Based on Lemma 3, there is a clause where $\text{val}(\mathcal{T}, m)$ as the head and the body is true under M .

1. $\mathcal{T} \neq \text{null}$. There is a rule where $\text{val}(\mathcal{T}, m)$ as the head, i.e., $\text{val}(\mathcal{T}, m) \leftarrow \text{val}(\mathcal{E}_1, m), \dots, \text{val}(\mathcal{E}_n, m)$. Then, we find that $M(\text{val}(\mathcal{E}_1, m) \wedge \dots \wedge \text{val}(\mathcal{E}_n, m)) = \top$. Therefore, $\text{val}(\mathcal{E}_i, m) \in M, 1 \leq i \leq n$. Based on Prop. 4, $\llbracket \mathcal{E}_i \rrbracket(\mathcal{Q}) = m, 1 \leq i \leq n$. Therefore, based on (4), we obtain $\llbracket \mathcal{T} \rrbracket(\mathcal{Q}) = m$.
2. $\mathcal{T} = \text{null}$. Then, there is a rule in Π where $\text{val}(\text{null}, m)$ as the head, i.e., $\text{val}(\text{null}, m) \leftarrow \top$. Thus, based on the definition (4), we obtain $\llbracket \mathcal{T} \rrbracket(\mathcal{Q}) = m$. \square

Lemma 14. Let $\Pi = \Pi_{\mathcal{Q}} \cup \Pi^{\mathcal{T}}$ be a program and M be an answer set of Π . Then,

$$\llbracket \mathcal{T} \rrbracket(\mathcal{Q}) = nm \quad \text{if and only if} \quad \text{val}(\mathcal{T}, nm) \in M.$$

Proof. (\Rightarrow) Suppose that $\llbracket \mathcal{T} \rrbracket(\mathcal{Q}) = nm$ holds. Then, as defined in (4), $\exists i : \llbracket \mathcal{E}_i \rrbracket(\mathcal{Q}) = nm$. Therefore, based on Prop. 4, $\exists i : \text{val}(\mathcal{E}_i, nm) \in M$. Hence, $\exists i : M(\text{val}(\mathcal{E}_i), nm) = \top$. Thus, by Lemma 1, $\text{val}(\mathcal{E}, nm) \in M$.

(\Leftarrow) Suppose that $\text{val}(\mathcal{T}, nm) \in M$. Based on Lemma 3, there is a clause where $\text{val}(\mathcal{T}, nm)$ as the head and the body is true under M . Based on AllOf transformation, $\exists i : M(\text{val}(\mathcal{E}_i), nm) = \top$. Therefore, $\exists i : \text{val}(\mathcal{E}_i, nm) \in M$. Based on Prop. 4, $\exists i : \llbracket \mathcal{E}_i \rrbracket(\mathcal{Q}) = nm$. Therefore, based on (4), we obtain $\llbracket \mathcal{T} \rrbracket(\mathcal{Q}) = nm$. \square

Lemma 15. Let $\Pi = \Pi_{\mathcal{Q}} \cup \Pi^{\mathcal{T}}$ be a program and M be an answer set of Π . Then,

$$\llbracket \mathcal{T} \rrbracket(\mathcal{Q}) = \text{idt} \quad \text{if and only if} \quad \text{val}(\mathcal{T}, \text{idt}) \in M.$$

Proof. (\Rightarrow) Suppose that $\llbracket \mathcal{T} \rrbracket(\mathcal{Q}) = \text{idt}$. Then, as defined in (4), $\llbracket \mathcal{T} \rrbracket(\mathcal{Q}) \neq m$ and $\llbracket \mathcal{T} \rrbracket(\mathcal{Q}) \neq nm$. Thus, by Lemma 13 and Lemma 14, $\text{val}(\mathcal{T}, m) \notin M$ and $\text{val}(\mathcal{T}, nm) \notin M$. Hence, $M(\text{not } \text{val}(\mathcal{T}, m) \wedge \text{not } \text{val}(\mathcal{T}, nm)) = \top$. Therefore, by Lemma 1, $\text{val}(\mathcal{T}, \text{idt}) \in M$.

(\Leftarrow) Suppose that $\text{val}(\mathcal{T}, \text{idt}) \in M$. Based on Lemma 3, there is a clause where $\text{val}(\mathcal{T}, \text{idt})$ as the head and the body is true under M . There is only one rule in Π with $\text{val}(\mathcal{T}, \text{idt})$ in the head, i.e., $\text{val}(\mathcal{T}, \text{idt}) \leftarrow \text{not } \text{val}(\mathcal{T}, m), \text{not } \text{val}(\mathcal{T}, nm)$. Thus, $\text{val}(\mathcal{T}, m) \notin M$ and $\text{val}(\mathcal{T}, nm) \notin M$. Based on Lemma 13 and Lemma 14, $\llbracket \mathcal{T} \rrbracket(\mathcal{Q}) \neq m$ and $\llbracket \mathcal{T} \rrbracket(\mathcal{Q}) \neq nm$. Therefore, based on (4) we obtain $\llbracket \mathcal{T} \rrbracket(\mathcal{Q}) = \text{idt}$. \square

Proposition 5. Let $\Pi = \Pi_{\mathcal{Q}} \cup \Pi^{\mathcal{T}}$ be a program obtained by merging Request transformation program $\Pi_{\mathcal{Q}}$ and Target \mathcal{T} transformations program with all of its components $\Pi^{\mathcal{T}}$. Let M be an answer set of Π . Then,

$$\llbracket \mathcal{T} \rrbracket(\mathcal{Q}) = V \quad \text{if and only if} \quad \text{val}(\mathcal{T}, V) \in M.$$

Proof. It follows from Lemma 13, Lemma 14 and Lemma 15 since the value of V only has three possibilities, i.e., $\{m, nm, \text{idt}\}$. \square

Condition Evaluation.

Proposition 6. *Let $\Pi = \Pi_Q \cup \Pi_C$ be a program obtained from merging Request transformation program Π_Q and Condition transformation program Π_C and let M be an answer set of Π . Then,*

$$\llbracket C \rrbracket(Q) = V \quad \text{if and only if} \quad \text{val}(C, V) \in M.$$

Proof. It follows from the equation (5) that the Condition evaluation based on the value of eval function, the same case in the Condition program transformation. \square

Rule Evaluation.

Lemma 16. *Let $\Pi = \Pi_Q \cup \Pi^R$ be a program obtained by merging Request transformation program Π_Q and Rule R transformations program with all of its components Π^R . Let M be an answer set of Π . Then,*

$$\llbracket R \rrbracket(Q) = E \quad \text{if and only if} \quad \text{val}(R, E) \in M$$

where E is Rule's effect, either p or d.

Proof. (\Rightarrow) Suppose that $\llbracket R \rrbracket(Q) = E$ holds. Then, as defined in (4), $\llbracket T \rrbracket(Q) = m$ and $\llbracket C \rrbracket(Q) = t$. Based on Prop. 7 and Prop. 6, $\text{val}(T, m) \in M$ and $\text{val}(C, t) \in M$. Thus, $M(\text{val}(T, m) \wedge \text{val}(C, t)) = \top$. Therefore, by Lemma 1, $\text{val}(R, E) \in M$.
 (\Leftarrow) Suppose that $\text{val}(R, E) \in M$. Based on Lemma 3, there is a clause where $\text{val}(R, E)$ as the head and the body is true under M . There is only one rule in Π with $\text{val}(R, E)$ in the head, i.e., $\text{val}(R, m) \leftarrow \text{val}(T, m), \text{val}(C, t)$. Then, we find that $M(\text{val}(T, m) \wedge \text{val}(C, t)) = \top$. Therefore, $\text{val}(T, m) \in M$ and $\text{val}(C, t) \in M$. Based on Prop. 5 and Prop. 6, $\llbracket T \rrbracket(Q) = m$ and $\llbracket C \rrbracket(Q) = t$. Therefore, based on (6) we obtain $\llbracket R \rrbracket(Q) = E$. \square

Lemma 17. *Let $\Pi = \Pi_Q \cup \Pi^R$ be a program obtained by merging Request transformation program Π_Q and Rule R transformations program with all of its components Π^R . Let M be an answer set of Π . Then,*

$$\llbracket R \rrbracket(Q) = na \quad \text{if and only if} \quad \text{val}(R, na) \in M.$$

Proof. (\Rightarrow) Suppose that $\llbracket R \rrbracket(Q) = na$ holds. Then, as defined in (6), we have that

1. $\llbracket T \rrbracket(Q) = m$ and $\llbracket C \rrbracket(Q) = f$. Based on Prop. 5 and Prop. 6, $\text{val}(T, m) \in M$ and $\text{val}(C, f) \in M$. Thus, $M(\text{val}(T, m) \wedge \text{val}(C, f)) = \top$. Therefore, by Lemma 1, $\text{val}(R, na) \in M$.
2. $\llbracket T \rrbracket(Q) = nm$. Based on Prop. 5, $\text{val}(T, nm) \in M$. Thus, $M(\text{val}(T, nm)) = \top$. Therefore, by Lemma 1, $\text{val}(R, na) \in M$.

(\Leftarrow) Suppose that $\text{val}(R, na) \in M$. Based on Lemma 3, there is a clause in Π where $\text{val}(R, na)$ as the head and the body is true under M . There are rules in Π where $\text{val}(R, na)$ as the head, i.e.,

1. $\text{val}(\mathcal{R}, \text{na}) \leftarrow \text{val}(\mathcal{T}, \text{m}), \text{val}(\mathcal{C}, \text{f})$.
Then, we find that $M(\text{val}(\mathcal{T}, \text{m}) \wedge \text{val}(\mathcal{C}, \text{f})) = \top$. Therefore, $\text{val}(\mathcal{T}, \text{m}) \in M$ and $\text{val}(\mathcal{C}, \text{f}) \in M$. Based on Prop. 5 and Prop. 6, $\llbracket \mathcal{T} \rrbracket(\mathcal{Q}) = \text{m}$ and $\llbracket \mathcal{C} \rrbracket(\mathcal{Q}) = \text{f}$. Therefore, based on (6), we obtain $\llbracket \mathcal{R} \rrbracket(\mathcal{Q}) = \text{na}$.
2. $\text{val}(\mathcal{R}, \text{na}) \leftarrow \text{val}(\mathcal{T}, \text{nm})$.
Then, we find that $M(\text{val}(\mathcal{T}, \text{nm})) = \top$. Therefore, $\text{val}(\mathcal{T}, \text{nm}) \in M$. Based on Prop. 5, $\llbracket \mathcal{T} \rrbracket(\mathcal{Q}) = \text{nm}$. Therefore, based on (6), we obtain $\llbracket \mathcal{R} \rrbracket(\mathcal{Q}) = \text{na}$. \square

Lemma 18. Let $\Pi = \Pi_{\mathcal{Q}} \cup \Pi^{\mathcal{R}}$ be a program and M be an answer set of Π . Then,

$$\llbracket \mathcal{R} \rrbracket(\mathcal{Q}) = i_E \quad \text{if and only if} \quad \text{val}(\mathcal{R}, i_E) \in M$$

where E is Rule's effect, either p or d .

Proof. (\Rightarrow) Suppose that $\llbracket \mathcal{R} \rrbracket(\mathcal{Q}) = i_E$. Then, as defined in (6), $\llbracket \mathcal{R} \rrbracket(\mathcal{Q}) \neq E$ and $\llbracket \mathcal{R} \rrbracket(\mathcal{Q}) \neq \text{na}$. By Lemma 16 and Lemma 17, $\text{val}(\mathcal{R}, E) \notin M$ and $\text{val}(\mathcal{R}, \text{na}) \notin M$. Hence, $M(\text{not val}(\mathcal{R}, E) \wedge \text{not val}(\mathcal{R}, \text{na})) = \top$. Thus, by Lemma 1, $\text{val}(\mathcal{R}, i_E) \in M$.

(\Leftarrow) Suppose that $\text{val}(\mathcal{R}, \text{idt}) \in M$. Based on Lemma 3, there is a clause where $\text{val}(\mathcal{R}, i_E)$ as the head and the body is true under M . There is only one rule in Π with $\text{val}(\mathcal{R}, i_E)$ in the head in, i.e., $\text{val}(\mathcal{R}, i_E) \leftarrow \text{not val}(\mathcal{R}, E), \text{not val}(\mathcal{R}, \text{na})$. Therefore, $M(\text{not val}(\mathcal{R}, E) \wedge \text{not val}(\mathcal{R}, \text{na})) = \top$. Thus, $\text{val}(\mathcal{R}, E) \notin M$ and $\text{val}(\mathcal{R}, \text{na}) \notin M$. Based on Lemma 16 and Lemma 17, $\llbracket \mathcal{R} \rrbracket(\mathcal{Q}) \neq E$ and $\llbracket \mathcal{R} \rrbracket(\mathcal{Q}) \neq \text{na}$. Hence, based on (6) we obtain, $\llbracket \mathcal{R} \rrbracket(\mathcal{Q}) = i_E$. \square

Proposition 7. Let $\Pi = \Pi_{\mathcal{Q}} \cup \Pi^{\mathcal{R}}$ be a program obtained by merging Request transformation program $\Pi_{\mathcal{Q}}$ and Rule \mathcal{R} transformations program with all of its components $\Pi^{\mathcal{R}}$. Let M be an answer set of Π . Then,

$$\llbracket \mathcal{R} \rrbracket(\mathcal{Q}) = V \quad \text{if and only if} \quad \text{val}(\mathcal{R}, V) \in M.$$

Proof. It follows from Lemma 16, Lemma 17 and Lemma 18 since the value of V only has five possibilities, i.e., $\{p, d, i_d, i_p, \text{na}\}$. \square

Combining Algorithm: Permit-Overrides.

Lemma 19. Let $\Pi = \Pi_{\mathcal{Q}} \cup \Pi_{\text{po}} \cup \Pi^{\mathcal{P}}$ be a program obtained by merging Request transformation program $\Pi_{\mathcal{Q}}$, permit-overrides combining algorithm transformation program Π_{po} and Policy \mathcal{P} transformation program with its components $\Pi^{\mathcal{P}}$. Let M be an answer set of Π . Then,

$$\bigoplus_{\text{po}}(\mathbf{R}) = p \quad \text{if and only if} \quad \text{algo}(\text{po}, \mathcal{P}, p) \in M$$

where $\mathbf{R} = \langle \llbracket \mathcal{R}_1 \rrbracket(\mathcal{Q}), \dots, \llbracket \mathcal{R}_n \rrbracket(\mathcal{Q}) \rangle$ be a sequence of policy value where each \mathcal{R}_i is a Rule in the sequence inside Policy \mathcal{P} .

Proof. (\Rightarrow) Suppose that $\bigoplus_{\text{po}}(\mathbf{R}) = p$ holds. Then, as defined in (8), $\exists i : \llbracket \mathcal{R}_i \rrbracket(\mathcal{Q}) = p$ where \mathcal{R}_i is a Rule in the sequence inside Policy \mathcal{P} . Based on Prop. 7, $\text{val}(\mathcal{R}_i, p) \in M$. Based on the Policy transformation, there is a rule in Π $\text{decision_of}(\mathcal{P}, \mathcal{R}_i, p) \leftarrow \text{val}(\mathcal{R}_i, p)$. Therefore, by Lemma 1, $\text{decision_of}(\mathcal{P}, \mathcal{R}_i, p) \in M$. Thus, by Lemma 1,

$\text{algo}(\text{po}, \mathcal{P}, \text{p}) \in M$.

(\Leftarrow) Suppose that $\text{algo}(\text{po}, \mathcal{P}, \text{p}) \in M$. Based on Lemma 3, there is a rule where $\text{algo}(\text{po}, \mathcal{P}, \text{p})$ as the head and the body is true under M . There is only one rule in Π with $\text{algo}(\text{po}, \mathcal{P}, \text{p})$ as the head, i.e., $\text{algo}(\text{po}, \mathcal{P}, \text{p}) \leftarrow \text{decision_of}(\mathcal{P}, \mathcal{R}, \text{p})$. Then, $M(\text{decision_of}(\mathcal{P}, \mathcal{R}, \text{p})) = \top$. Therefore, $\text{decision_of}(\mathcal{P}, \mathcal{R}, \text{p}) \in M$. Based on Lemma 3, there is a rule where $\text{decision_of}(\mathcal{P}, \mathcal{R}, \text{p})$ as the head and the body is true under M . There is only one rule in Π , i.e., $\text{decision_of}(\mathcal{P}, \mathcal{R}, \text{p}) \leftarrow \text{val}(\mathcal{R}, \text{p})$. Then, $M(\text{val}(\mathcal{R}, \text{p})) = \top$. Therefore, $\text{val}(\mathcal{R}, \text{p}) \in M$. Based on Prop. 7, $\llbracket \mathcal{R} \rrbracket(\mathcal{Q}) = \text{p}$ and \mathcal{R} belongs to the sequence inside Policy \mathcal{P} . Therefore, based on (8), we obtain $\bigoplus_{\text{po}}(\mathbf{R}) = \text{p}$ \square

Lemma 20. Let $\Pi = \Pi_{\mathcal{Q}} \cup \Pi_{\text{po}} \cup \Pi^{\mathcal{P}}$ be a program obtained by merging Request transformation program $\Pi_{\mathcal{Q}}$, permit-overrides combining algorithm transformation program Π_{po} and Policy \mathcal{P} transformation program with its components $\Pi^{\mathcal{P}}$. Let M be an answer set of Π . Then,

$$\bigoplus_{\text{po}}(\mathbf{R}) = \text{idp} \quad \text{if and only if} \quad \text{algo}(\text{po}, \mathcal{P}, \text{idp}) \in M$$

where $\mathbf{R} = \langle \llbracket \mathcal{R}_1 \rrbracket(\mathcal{Q}), \dots, \llbracket \mathcal{R}_n \rrbracket(\mathcal{Q}) \rangle$ be a sequence of policy value where each \mathcal{R}_i is a Rule in the sequence inside Policy \mathcal{P} .

Proof. (\Rightarrow) Suppose that $\bigoplus_{\text{po}}(\mathbf{R}) = \text{idp}$ holds. Then, as defined in (8) we have that

1. $\forall i : \llbracket \mathcal{R}_i \rrbracket(\mathcal{Q}) \neq \text{p}$ and $\exists j : \llbracket \mathcal{R}_j \rrbracket(\mathcal{Q}) = \text{idp}$ where \mathcal{R}_i and \mathcal{R}_j are Rule in the sequence inside Policy \mathcal{P} . Based on Prop. 7, $\forall i : \text{val}(\mathcal{R}_i, \text{p}) \notin M$ and $\exists j : \text{val}(\mathcal{R}_j, \text{idp}) \in M$. Based on Lemma 19, $\text{algo}(\text{po}, \mathcal{P}, \text{p}) \notin M$ since if it is in M , there exists a Rule \mathcal{R} in the Policy \mathcal{P} sequence such that $\llbracket \mathcal{R} \rrbracket(\mathcal{Q}) = \text{p}$. Based on the Policy transformation, there is a rule $\text{decision_of}(\mathcal{P}, \mathcal{R}_j, \text{idp}) \leftarrow \text{val}(\mathcal{R}_j, \text{idp})$. By Lemma 1, $\text{decision_of}(\mathcal{P}, \mathcal{R}_j, \text{idp}) \in M$. Thus, by Lemma 1, $\text{algo}(\text{po}, \mathcal{P}, \text{idp}) \in M$.
2. $\forall i : \llbracket \mathcal{R}_i \rrbracket(\mathcal{Q}) \neq \text{p}$ and $\exists j : \llbracket \mathcal{R}_j \rrbracket(\mathcal{Q}) = \text{idp}$ and $\exists j' : \llbracket \mathcal{R}_{j'} \rrbracket(\mathcal{Q}) = \text{d}$ where \mathcal{R}_i , \mathcal{R}_j and $\mathcal{R}_{j'}$ are Rules in the sequence inside Policy \mathcal{P} . Based on Prop. 7, $\forall i : \text{val}(\mathcal{R}_i, \text{p}) \notin M$ and $\exists j : \text{val}(\mathcal{R}_j, \text{idp}) \in M$ and $\exists j' : \text{val}(\mathcal{R}_{j'}, \text{d}) \in M$. Based on Lemma 19, $\text{algo}(\text{po}, \mathcal{P}, \text{p}) \notin M$ since if it is in M , there exists a Rule \mathcal{R} in the Policy \mathcal{P} sequence such that $\llbracket \mathcal{R} \rrbracket(\mathcal{Q}) = \text{p}$. Based on the Policy transformation, there are rules in Π in the form $\text{decision_of}(\mathcal{P}, \mathcal{R}_j, \text{idp}) \leftarrow \text{val}(\mathcal{R}_j, \text{idp})$ and $\text{decision_of}(\mathcal{P}, \mathcal{R}_{j'}, \text{d}) \leftarrow \text{val}(\mathcal{R}_{j'}, \text{d})$. Thus, by Lemma 1, $\text{decision_of}(\mathcal{P}, \mathcal{R}_j, \text{idp}) \in M$ and $\text{decision_of}(\mathcal{P}, \mathcal{R}_{j'}, \text{d}) \in M$. Hence, by Lemma 1, $\text{algo}(\text{po}, \mathcal{P}, \text{idp}) \in M$.
3. $\forall i : \llbracket \mathcal{R}_i \rrbracket(\mathcal{Q}) \neq \text{p}$ and $\exists j : \llbracket \mathcal{R}_j \rrbracket(\mathcal{Q}) = \text{idp}$ and $\exists j' : \llbracket \mathcal{R}_{j'} \rrbracket(\mathcal{Q}) = \text{id}$ where \mathcal{R}_i , \mathcal{R}_j and $\mathcal{R}_{j'}$ are Rule in the sequence inside Policy \mathcal{P} . Based on Prop. 7, $\forall i : \text{val}(\mathcal{R}_i, \text{p}) \notin M$ and $\exists j : \text{val}(\mathcal{R}_j, \text{idp}) \in M$ and $\exists j' : \text{val}(\mathcal{R}_{j'}, \text{id}) \in M$. Based on Lemma 19, $\text{algo}(\text{po}, \mathcal{P}, \text{p}) \notin M$ since if it is in M , there exists a Rule \mathcal{R} in the Policy \mathcal{P} sequence such that $\llbracket \mathcal{R} \rrbracket(\mathcal{Q}) = \text{p}$. Based on the Policy transformation there are rules in Π in the form $\text{decision_of}(\mathcal{P}, \mathcal{R}_j, \text{idp}) \leftarrow \text{val}(\mathcal{R}_j, \text{idp})$ and $\text{decision_of}(\mathcal{P}, \mathcal{R}_{j'}, \text{id}) \leftarrow \text{val}(\mathcal{R}_{j'}, \text{id})$. Thus, by Lemma 1, $\text{decision_of}(\mathcal{P}, \mathcal{R}_j, \text{idp}) \in M$ and $\text{decision_of}(\mathcal{P}, \mathcal{R}_{j'}, \text{id}) \in M$. Hence, by Lemma 1, $\text{algo}(\text{po}, \mathcal{P}, \text{idp}) \in M$.

(\Leftarrow) Suppose that $\text{algo}(\text{po}, \mathcal{P}, \text{idp}) \in M$. Based on Lemma 3, there is a rule where $\text{algo}(\text{po}, \mathcal{P}, \text{idp})$ as the head and the body is true under M . There are rules in Π where $\text{algo}(\text{po}, \mathcal{P}, \text{idp})$ as the head, i.e.,

1. $\text{algo}(\text{po}, \mathcal{P}, i_{dp}) \leftarrow \text{not } \text{algo}(\text{po}, \mathcal{P}, p), \text{decision_of}(\mathcal{P}, \mathcal{R}, i_{dp})$.
Then, $M(\text{not } \text{algo}(\text{po}, \mathcal{P}, p) \wedge \text{decision_of}(\mathcal{P}, \mathcal{R}, i_{dp})) = \top$. Thus, $\text{algo}(\text{po}, \mathcal{P}, p) \notin M$ and $\text{decision_of}(\mathcal{P}, \mathcal{R}, i_{dp}) \in M$. Based on Lemma 19, $\bigoplus_{po}(\mathbf{R}) \neq p$. Based on Lemma 3, there is a rule where $\text{decision_of}(\mathcal{P}, \mathcal{R}, i_{dp})$ as the head and the body is true under M . There is only one rule in Π , i.e., $\text{decision_of}(\mathcal{P}, \mathcal{R}, i_{dp}) \leftarrow \text{val}(\mathcal{R}, i_{dp})$. Then, $M(\text{val}(\mathcal{R}, i_{dp})) = \top$. Therefore, $\text{val}(\mathcal{R}, i_{dp}) \in M$. As defined in (8), $\forall i : \llbracket \mathcal{R}_i \rrbracket(\mathcal{Q}) \neq p$ since $\bigoplus_{po}(\mathbf{R}) \neq p$. Based on Prop. 7, $\llbracket \mathcal{R} \rrbracket(\mathcal{Q}) = i_{dp}$ and \mathcal{R} belongs to the sequence inside Policy \mathcal{P} . Hence, based on (8), we obtain $\bigoplus_{po}(\mathbf{R}) = i_{dp}$.
2. $\text{algo}(\text{po}, \mathcal{P}, i_{dp}) \leftarrow \text{not } \text{algo}(\text{po}, \mathcal{P}, p), \text{decision_of}(\mathcal{P}, \mathcal{R}, i_p), \text{decision_of}(\mathcal{P}, \mathcal{R}', d)$.
Then, $M(\text{not } \text{algo}(\text{po}, \mathcal{P}, p) \wedge \text{decision_of}(\mathcal{P}, \mathcal{R}, i_p) \wedge \text{decision_of}(\mathcal{P}, \mathcal{R}', d)) = \top$. Thus, $\text{algo}(\text{po}, \mathcal{P}, p) \notin M$, $\text{decision_of}(\mathcal{P}, \mathcal{R}, i_p) \in M$ and $\text{decision_of}(\mathcal{P}, \mathcal{R}, d) \in M$. Based on Lemma 19, $\bigoplus_{po}(\mathbf{R}) \neq p$. Based on Lemma 3, there is a rule where $\text{decision_of}(\mathcal{P}, \mathcal{R}, i_p)$ as the head and the body is true under M . There is only one rule in Π , i.e., $\text{decision_of}(\mathcal{P}, \mathcal{R}, i_p) \leftarrow \text{val}(\mathcal{R}, i_p)$. Then, $M(\text{val}(\mathcal{R}, i_p)) = \top$. Thus, $\text{val}(\mathcal{R}, i_p) \in M$. Based on Lemma 3, there is a rule where $\text{decision_of}(\mathcal{P}, \mathcal{R}, d)$ as the head and the body is true under M . There is only one rule in Π , i.e., $\text{decision_of}(\mathcal{P}, \mathcal{R}, i_p) \leftarrow \text{val}(\mathcal{R}', d)$. Then, $M(\text{val}(\mathcal{R}', d)) = \top$. Thus, $\text{val}(\mathcal{R}', d) \in M$. Based on (8), $\forall i : \llbracket \mathcal{R}_i \rrbracket(\mathcal{Q}) \neq p$ since $\bigoplus_{po}(\mathbf{R}) \neq p$. Based on Prop. 7, $\llbracket \mathcal{R} \rrbracket(\mathcal{Q}) = i_p$ and $\llbracket \mathcal{R}' \rrbracket(\mathcal{Q}) = d$ and $\mathcal{R}, \mathcal{R}'$ belongs to the sequence inside Policy \mathcal{P} . Therefore, based on (8) we obtain $\bigoplus_{po}(\mathbf{R}) = i_{dp}$.
3. $\text{algo}(\text{po}, \mathcal{P}, i_{dp}) \leftarrow \text{not } \text{algo}(\text{po}, \mathcal{P}, p), \text{decision_of}(\mathcal{P}, \mathcal{R}, i_p), \text{decision_of}(\mathcal{P}, \mathcal{R}', i_d)$.
Then, $M(\text{not } \text{algo}(\text{po}, \mathcal{P}, p) \wedge \text{decision_of}(\mathcal{P}, \mathcal{R}, i_p) \wedge \text{decision_of}(\mathcal{P}, \mathcal{R}', i_d)) = \top$. Thus, $\text{algo}(\text{po}, \mathcal{P}, p) \notin M$, $\text{decision_of}(\mathcal{P}, \mathcal{R}, i_p) \in M$ and $\text{decision_of}(\mathcal{P}, \mathcal{R}, i_d) \in M$. Based on Lemma 19, $\bigoplus_{po}(\mathbf{R}) \neq p$ since if $\bigoplus_{po}(\mathbf{R}) = p$. Based on Lemma 3, there is a rule where $\text{decision_of}(\mathcal{P}, \mathcal{R}, i_p)$ as the head and the body is true under M . There is only one rule in Π , i.e., $\text{decision_of}(\mathcal{P}, \mathcal{R}, i_p) \leftarrow \text{val}(\mathcal{R}, i_p)$. Then, $M(\text{val}(\mathcal{R}, i_p)) = \top$. Therefore, $\text{val}(\mathcal{R}, i_p) \in M$. Based on Lemma 3, there is a rule where $\text{decision_of}(\mathcal{P}, \mathcal{R}, i_d)$ as the head and the body is true under M . There is only one rule in Π , i.e., $\text{decision_of}(\mathcal{P}, \mathcal{R}, i_p) \leftarrow \text{val}(\mathcal{R}', i_d)$. Then, $M(\text{val}(\mathcal{R}', i_d)) = \top$. Therefore, $\text{val}(\mathcal{R}', i_d) \in M$. Based on (8), $\forall i : \llbracket \mathcal{R}_i \rrbracket(\mathcal{Q}) \neq p$ since $\bigoplus_{po}(\mathbf{R}) \neq p$. Based on Prop. 7, $\llbracket \mathcal{R} \rrbracket(\mathcal{Q}) = i_p$ and $\llbracket \mathcal{R}' \rrbracket(\mathcal{Q}) = i_d$ and $\mathcal{R}, \mathcal{R}'$ belongs to the sequence inside Policy \mathcal{P} . Therefore, based on (8), we obtain $\bigoplus_{po}(\mathbf{R}) = i_{dp}$ \square

Lemma 21. Let $\Pi = \Pi_{\mathcal{Q}} \cup \Pi_{po} \cup \Pi^{\mathcal{P}}$ be a program obtained by merging Request transformation program $\Pi_{\mathcal{Q}}$, permit-overrides combining algorithm transformation program Π_{po} and Policy \mathcal{P} transformation program with its components $\Pi^{\mathcal{P}}$. Let M be an answer set of Π . Then,

$$\bigoplus_{po}(\mathbf{R}) = i_p \quad \text{if and only if} \quad \text{algo}(\text{po}, \mathcal{P}, i_p) \in M$$

where $\mathbf{R} = \langle \llbracket \mathcal{R}_1 \rrbracket(\mathcal{Q}), \dots, \llbracket \mathcal{R}_n \rrbracket(\mathcal{Q}) \rangle$ be a sequence of policy value where each \mathcal{R}_i is a Rule in the sequence inside Policy \mathcal{P} .

Proof. (\Rightarrow) Suppose that $\bigoplus_{po}(\mathbf{R}) = i_p$ holds. Then, as defined in (8), $\exists i : \llbracket \mathcal{R}_i \rrbracket(\mathcal{Q}) = i_p$ and $\forall j : \llbracket \mathcal{R}_j \rrbracket(\mathcal{Q}) \neq i_p \Rightarrow \llbracket \mathcal{R}_j \rrbracket(\mathcal{Q}) = \text{na}$ where \mathcal{R}_i and \mathcal{R}_j are Rule in the sequence inside Policy \mathcal{P} . Based on Prop. 7, $\exists i : \text{val}(\mathcal{R}_i, i_p) \in M$. Based on Lemma 19,

$\text{algo}(\text{po}, \mathcal{P}, p) \notin M$ since if it is in M , there exists a Rule \mathcal{R} in the Policy \mathcal{P} sequence such that $\llbracket \mathcal{R} \rrbracket(\mathcal{Q}) = p$. Based on Lemma 20, $\text{algo}(\text{po}, \mathcal{P}, i_{dp}) \notin M$ since if it is in M , there exists a Rule \mathcal{R} in the Policy \mathcal{P} sequence such that $\llbracket \mathcal{R} \rrbracket(\mathcal{Q}) = i_{dp}$, and $\llbracket \mathcal{R} \rrbracket(\mathcal{Q}) = d$ or i_d . Based on the Policy transformation, there is a rule $\text{decision_of}(\mathcal{P}, \mathcal{R}_i, i_p) \leftarrow \text{val}(\mathcal{R}_i, i_p)$. Therefore, by Lemma 1, $\text{decision_of}(\mathcal{P}, \mathcal{R}_i, i_p) \in M$. Thus, by Lemma 1, $\text{algo}(\text{po}, \mathcal{P}, i_p) \in M$.

(\Leftarrow) Suppose that $\text{algo}(\text{po}, \mathcal{P}, i_p) \in M$. Based on Lemma 3, there is a rule where $\text{algo}(\text{po}, \mathcal{P}, i_p)$ as the head and the body is true under M . There is only a rule in Π , i.e., $\text{algo}(\text{po}, \mathcal{P}, i_{dp}) \leftarrow \text{not } \text{algo}(\text{po}, \mathcal{P}, p), \text{not } \text{algo}(\text{po}, \mathcal{P}, i_{dp}), \text{decision_of}(\mathcal{P}, \mathcal{R}, i_{dp})$. Then, $M(\text{not } \text{algo}(\text{po}, \mathcal{P}, p) \wedge \text{not } \text{algo}(\text{po}, \mathcal{P}, i_{dp}) \wedge \text{decision_of}(\mathcal{P}, \mathcal{R}, i_{dp})) = \top$. Therefore, $\text{algo}(\text{po}, \mathcal{P}, p) \notin M$, $\text{algo}(\text{po}, \mathcal{P}, i_{dp}) \notin M$ and $\text{decision_of}(\mathcal{P}, \mathcal{R}, i_{dp}) \in M$. Based on Lemma 19 and Lemma 20, $\bigoplus_{po}(\mathbf{R}) \neq p$ and $\bigoplus_{po}(\mathbf{R}) \neq i_{dp}$. Based on Lemma 3, there is a rule where $\text{decision_of}(\mathcal{P}, \mathcal{R}, i_p)$ as the head and the body is true under M . There is only one rule in Π , i.e., $\text{decision_of}(\mathcal{P}, \mathcal{R}, i_p) \leftarrow \text{val}(\mathcal{R}, i_p)$. Then, $M(\text{val}(\mathcal{R}, i_p)) = \top$. Therefore, $\text{val}(\mathcal{R}, i_p) \in M$. Based on (8), $\forall i : \llbracket \mathcal{R}_i \rrbracket(\mathcal{Q}) \neq p$ since $\bigoplus_{po}(\mathbf{R}) \neq p$ and $\forall i : \llbracket \mathcal{R}_i \rrbracket(\mathcal{Q}) \neq (i_{dp} \text{ or } d \text{ or } i_d)$. Thus, the only possibilities of the value of $\llbracket \mathcal{R}_i \rrbracket$ is either i_p or na. Based on Prop. 7, $\llbracket \mathcal{R} \rrbracket(\mathcal{Q}) = i_p$ and \mathcal{R} belongs to the sequence inside Policy \mathcal{P} . Therefore, based on (8) we obtain $\bigoplus_{po}(\mathbf{R}) = i_p$ \square

Lemma 22. Let $\Pi = \Pi_{\mathcal{Q}} \cup \Pi_{po} \cup \Pi^P$ be a program obtained by merging Request transformation program $\Pi_{\mathcal{Q}}$, permit-overrides combining algorithm transformation program Π_{po} and Policy \mathcal{P} transformation program with its components Π^P . Let M be an answer set of Π . Then,

$$\bigoplus_{po}(\mathbf{R}) = d \quad \text{if and only if} \quad \text{algo}(\text{po}, \mathcal{P}, d) \in M$$

where $\mathbf{R} = \langle \llbracket \mathcal{R}_1 \rrbracket(\mathcal{Q}), \dots, \llbracket \mathcal{R}_n \rrbracket(\mathcal{Q}) \rangle$ be a sequence of policy value where each \mathcal{R}_i is a Rule in the sequence inside Policy \mathcal{P} .

Proof. (\Rightarrow) Suppose that $\bigoplus_{po}(\mathbf{R}) = d$ holds. Then, as defined in (8), $\exists i : \llbracket \mathcal{R}_i \rrbracket(\mathcal{Q}) = d$ and $\forall j : \llbracket \mathcal{R}_j \rrbracket(\mathcal{Q}) \neq d \Rightarrow \llbracket \mathcal{R}_j \rrbracket(\mathcal{Q}) = (i_d \text{ or } na)$ where \mathcal{R}_i and \mathcal{R}_j are Rule in the sequence inside Policy \mathcal{P} . Based on Prop. 7, $\exists i : \text{val}(\mathcal{R}_i, d) \in M$. Based on Lemma 19, $\text{algo}(\text{po}, \mathcal{P}, p) \notin M$ since if it is in M , there exists a Rule \mathcal{R} in the Policy \mathcal{P} sequence such that $\llbracket \mathcal{R} \rrbracket(\mathcal{Q}) = p$. Based on Lemma 20, $\text{algo}(\text{po}, \mathcal{P}, i_{dp}) \notin M$ since if it is in M , there exists a Rule \mathcal{R} in the Policy \mathcal{P} sequence such that $\llbracket \mathcal{R} \rrbracket(\mathcal{Q}) = i_{dp}$. Based on Lemma 21, $\text{algo}(\text{po}, \mathcal{P}, i_p) \notin M$ since if it is in M , there exists a Rule \mathcal{R} in the Policy \mathcal{P} sequence such that $\llbracket \mathcal{R} \rrbracket(\mathcal{Q}) = i_p$. Based on the Policy transformation, there is a rule $\text{decision_of}(\mathcal{P}, \mathcal{R}_i, d) \leftarrow \text{val}(\mathcal{R}_i, d)$. Therefore, by Lemma 1, $\text{decision_of}(\mathcal{P}, \mathcal{R}_i, d) \in M$. Thus, by Lemma 1, $\text{algo}(\text{po}, \mathcal{P}, i_d) \in M$.

(\Leftarrow) Suppose that $\text{algo}(\text{po}, \mathcal{P}, d) \in M$. Based on Lemma 3, there is a rule where $\text{algo}(\text{po}, \mathcal{P}, d)$ as the head and the body is true under M . There is only a rule in Π , i.e., $\text{algo}(\text{po}, \mathcal{P}, i_{dp}) \leftarrow \text{not } \text{algo}(\text{po}, \mathcal{P}, p), \text{not } \text{algo}(\text{po}, \mathcal{P}, i_{dp}), \text{not } \text{algo}(\text{po}, \mathcal{P}, i_p), \text{decision_of}(\mathcal{P}, \mathcal{R}, d)$. Hence, we obtain $M(\text{not } \text{algo}(\text{po}, \mathcal{P}, p) \wedge \text{not } \text{algo}(\text{po}, \mathcal{P}, i_{dp}) \wedge \text{not } \text{algo}(\text{po}, \mathcal{P}, i_p) \wedge \text{decision_of}(\mathcal{P}, \mathcal{R}, d)) = \top$. Therefore, $\text{algo}(\text{po}, \mathcal{P}, p) \notin M$, $\text{algo}(\text{po}, \mathcal{P}, i_{dp}) \notin M$, $\text{algo}(\text{po}, \mathcal{P}, i_p) \notin M$ and $\text{decision_of}(\mathcal{P}, \mathcal{R}, d) \in M$. Based on Lemma 19, $\bigoplus_{po}(\mathbf{R}) \neq p$ since if $\bigoplus_{po}(\mathbf{R}) = p$ it will lead a contradiction. Based on Lemma 20, $\bigoplus_{po}(\mathbf{R}) \neq i_{dp}$ since if $\bigoplus_{po}(\mathbf{R}) = i_{dp}$ it will lead a contradiction. Based on Lemma 21, $\bigoplus_{po}(\mathbf{R}) \neq i_p$ since if $\bigoplus_{po}(\mathbf{R}) = i_p$ it will lead a contradiction. Based

on Lemma 3, there is a rule where $\text{decision_of}(\mathcal{P}, \mathcal{R}, i_d)$ as the head and the body is true under M . There is only one rule in Π , i.e., $\text{decision_of}(\mathcal{P}, \mathcal{R}, d) \leftarrow \text{val}(\mathcal{R}, d)$. Then, $M(\text{val}(\mathcal{R}, d)) = \top$. Therefore, $\text{val}(\mathcal{R}, d) \in M$. Based on (8), $\forall i : \llbracket \mathcal{R}_i \rrbracket(\mathcal{Q}) \neq p$ since $\bigoplus_{po}(\mathbf{R}) \neq p$. Based on (8), $\forall i : \llbracket \mathcal{R}_i \rrbracket(\mathcal{Q}) \neq i_{dp}$. Based on (8), $\forall i : \llbracket \mathcal{R}_i \rrbracket(\mathcal{Q}) \neq i_p$. Thus, the only possibilities of the value of $\llbracket \mathcal{R}_i \rrbracket$ is either d, i_d or na . Based on Prop. 7, $\llbracket \mathcal{R} \rrbracket(\mathcal{Q}) = d$ and \mathcal{R} belongs to the sequence inside Policy \mathcal{P} . Therefore, based on (8), $\bigoplus_{po}(\mathbf{R}) = d$ \square

Lemma 23. Let $\Pi = \Pi_Q \cup \Pi_{po} \cup \Pi^P$ be a program obtained by merging Request transformation program Π_Q , permit-overrides combining algorithm transformation program Π_{po} and Policy \mathcal{P} transformation program with its components Π^P . Let M be an answer set of Π . Then,

$$\bigoplus_{po}(\mathbf{R}) = i_d \quad \text{if and only if} \quad \text{algo}(po, \mathcal{P}, i_d) \in M$$

where $\mathbf{R} = \langle \llbracket \mathcal{R}_1 \rrbracket(\mathcal{Q}), \dots, \llbracket \mathcal{R}_n \rrbracket(\mathcal{Q}) \rangle$ be a sequence of policy value where each \mathcal{R}_i is a Rule in the sequence inside Policy \mathcal{P} .

Proof. (\Rightarrow) Suppose that $\bigoplus_{po}(\mathbf{R}) = i_d$ holds. Then, as defined in (8) $\exists i : \llbracket \mathcal{R}_i \rrbracket(\mathcal{Q}) = i_d$ and $\forall j : \llbracket \mathcal{R}_j \rrbracket(\mathcal{Q}) \neq d \Rightarrow \llbracket \mathcal{R}_j \rrbracket(\mathcal{Q}) = na$ where \mathcal{R}_i and \mathcal{R}_j are Rule in the sequence inside Policy \mathcal{P} . Based on Prop. 7, $\exists i : \text{val}(\mathcal{R}_i, d) \in M$. Based on Lemma 19, $\text{algo}(po, \mathcal{P}, p) \notin M$ since if it is in M , there exists a Rule \mathcal{R} in the Policy \mathcal{P} sequence such that $\llbracket \mathcal{R} \rrbracket(\mathcal{Q}) = p$. Based on Lemma 20, $\text{algo}(po, \mathcal{P}, i_{dp}) \notin M$ since if it is in M , there exists a Rule \mathcal{R} in the Policy \mathcal{P} sequence such that $\llbracket \mathcal{R} \rrbracket(\mathcal{Q}) = i_{dp}$. Based on Lemma 21, $\text{algo}(po, \mathcal{P}, i_p) \notin M$ since if it is in M , there exists a Rule \mathcal{R} in the Policy \mathcal{P} sequence such that $\llbracket \mathcal{R} \rrbracket(\mathcal{Q}) = i_p$. Based on Lemma 22, $\text{algo}(po, \mathcal{P}, d) \notin M$ since if it is in M , there exists a Rule \mathcal{R} in the Policy \mathcal{P} sequence such that $\llbracket \mathcal{R} \rrbracket(\mathcal{Q}) = d$. Based on the Policy transformation, there is a rule $\text{decision_of}(\mathcal{P}, \mathcal{R}_i, i_d) \leftarrow \text{val}(\mathcal{R}_i, i_d)$. Hence, by Lemma 1, $\text{decision_of}(\mathcal{P}, \mathcal{R}_i, i_d) \in M$. Thus, by Lemma 1, $\text{algo}(po, \mathcal{P}, i_d) \in M$. (\Leftarrow) Suppose that $\text{algo}(po, \mathcal{P}, d) \in M$. Based on Lemma 3, there is a rule where $\text{algo}(po, \mathcal{P}, d)$ as the head and the body is true under M . There is only a rule in Π , i.e., $\text{algo}(po, \mathcal{P}, d) \leftarrow \text{not algo}(po, \mathcal{P}, p), \text{not algo}(po, \mathcal{P}, i_{dp}), \text{not algo}(po, \mathcal{P}, i_p), \text{not algo}(po, \mathcal{P}, d), \text{decision_of}(\mathcal{P}, \mathcal{R}, i_d)$. Hence, we find that $M(\text{not algo}(po, \mathcal{P}, p) \wedge \text{not algo}(po, \mathcal{P}, i_{dp}) \wedge \text{not algo}(po, \mathcal{P}, i_p) \wedge \text{not algo}(po, \mathcal{P}, d) \wedge \text{decision_of}(\mathcal{P}, \mathcal{R}, i_d)) = \top$. Thus, $\text{algo}(po, \mathcal{P}, p) \notin M, \text{algo}(po, \mathcal{P}, i_{dp}) \notin M, \text{algo}(po, \mathcal{P}, i_p) \notin M, \text{algo}(po, \mathcal{P}, d) \notin M$ and $\text{decision_of}(\mathcal{P}, \mathcal{R}, i_d) \in M$. Based on Lemma 19, $\bigoplus_{po}(\mathbf{R}) \neq p$ since if $\bigoplus_{po}(\mathbf{R}) = p$ it will lead a contradiction. Based on Lemma 20, $\bigoplus_{po}(\mathbf{R}) \neq i_{dp}$ since if $\bigoplus_{po}(\mathbf{R}) = i_{dp}$ it will lead a contradiction. Based on Lemma 21, $\bigoplus_{po}(\mathbf{R}) \neq i_p$ since if $\bigoplus_{po}(\mathbf{R}) = i_p$ it will lead a contradiction. Based on Lemma 22, $\bigoplus_{po}(\mathbf{R}) \neq d$ since if $\bigoplus_{po}(\mathbf{R}) = d$ it will lead a contradiction. Based on Lemma 3, there is a rule where $\text{decision_of}(\mathcal{P}, \mathcal{R}, d)$ as the head and the body is true under M . There is only one rule in Π , i.e., $\text{decision_of}(\mathcal{P}, \mathcal{R}, i_d) \leftarrow \text{val}(\mathcal{R}, i_d)$. Then, we find that $M(\text{val}(\mathcal{R}, i_d)) = \top$. Therefore, $\text{val}(\mathcal{R}, i_d) \in M$. Based on (8), $\forall i : \llbracket \mathcal{R}_i \rrbracket(\mathcal{Q}) \neq p$ since $\bigoplus_{po}(\mathbf{R}) \neq p$. Based on eqrefeq:po, $\forall i : \llbracket \mathcal{R}_i \rrbracket(\mathcal{Q}) \neq i_{dp}$. Based on (8), $\forall i : \llbracket \mathcal{R}_i \rrbracket(\mathcal{Q}) \neq i_p$ and $\forall i : \llbracket \mathcal{R}_i \rrbracket(\mathcal{Q}) \neq d$. Thus, the only possibilities of the value of $\llbracket \mathcal{R}_i \rrbracket$ is either i_d or na . Based on Prop. 7, $\llbracket \mathcal{R} \rrbracket(\mathcal{Q}) = i_d$ and \mathcal{R} belongs to the sequence inside Policy \mathcal{P} . Therefore, based on (8), $\bigoplus_{po}(\mathbf{R}) = i_d$ \square

Lemma 24. Let $\Pi = \Pi_Q \cup \Pi_{po} \cup \Pi^P$ be a program obtained by merging Request transformation program Π_Q , permit-overrides combining algorithm transformation program Π_{po} and Policy \mathcal{P} transformation program with its components Π^P . Let M be an answer set of Π . Then,

$$\bigoplus_{po}(\mathbf{R}) = na \quad \text{if and only if} \quad algo(po, \mathcal{P}, na) \in M$$

where $\mathbf{R} = \langle \llbracket \mathcal{R}_1 \rrbracket(\mathcal{Q}), \dots, \llbracket \mathcal{R}_n \rrbracket(\mathcal{Q}) \rangle$ be a sequence of policy value where each \mathcal{R}_i is a Rule in the sequence inside Policy \mathcal{P} .

Proof. (\Rightarrow) Suppose that $\bigoplus_{po}(\mathbf{R}) = na$ holds. Then, as defined in (8) we have that $\bigoplus_{po}(\mathbf{R}) \neq p$, $\bigoplus_{po}(\mathbf{R}) \neq idp$, $\bigoplus_{po}(\mathbf{R}) \neq ip$, $\bigoplus_{po}(\mathbf{R}) \neq d$, and $\bigoplus_{po}(\mathbf{R}) \neq id$. Based on Lemma 19, $algo(po, \mathcal{P}, p) \notin M$. Based on Lemma 20, $algo(po, \mathcal{P}, idp) \notin M$. Based on Lemma 21, $algo(po, \mathcal{P}, ip) \notin M$. Based on Lemma 22, $algo(po, \mathcal{P}, d) \notin M$. Based on Lemma 23, $algo(po, \mathcal{P}, id) \notin M$. Thus, $M(\text{not } algo(po, \mathcal{P}, p) \wedge \text{not } algo(po, \mathcal{P}, idp) \wedge \text{not } algo(po, \mathcal{P}, ip) \wedge \text{not } algo(po, \mathcal{P}, d) \wedge \text{not } algo(po, \mathcal{P}, id)) = \top$. Therefore, by Lemma 1, $algo(po, \mathcal{P}, na) \in M$.

(\Leftarrow) Suppose that $algo(po, \mathcal{P}, na) \in M$. Based on Lemma 3, there is a rule where $algo(po, \mathcal{P}, na)$ as the head and the body is true under M . There is only a rule in Π where $algo(po, \mathcal{P}, na)$ as the head, i.e., $algo(po, \mathcal{P}, na) \leftarrow \text{not } algo(po, \mathcal{P}, p), \text{not } algo(po, \mathcal{P}, idp), \text{not } algo(po, \mathcal{P}, ip), \text{not } algo(po, \mathcal{P}, d), \text{not } algo(po, \mathcal{P}, id)$. Then, $M(\text{not } algo(po, \mathcal{P}, p) \wedge \text{not } algo(po, \mathcal{P}, idp) \wedge \text{not } algo(po, \mathcal{P}, ip) \wedge \text{not } algo(po, \mathcal{P}, d) \wedge \text{not } algo(po, \mathcal{P}, id)) = \top$. Therefore, $algo(po, \mathcal{P}, p) \notin M$, $algo(po, \mathcal{P}, idp) \notin M$, $algo(po, \mathcal{P}, ip) \notin M$, $algo(po, \mathcal{P}, d) \notin M$ and $algo(po, \mathcal{P}, id) \notin M$. Based on Lemma 19, $\bigoplus_{po}(\mathbf{R}) \neq p$. Based on Lemma 20, $\bigoplus_{po}(\mathbf{R}) \neq idp$. Based on Lemma 21, $\bigoplus_{po}(\mathbf{R}) \neq ip$. Based on Lemma 22, $\bigoplus_{po}(\mathbf{R}) \neq d$. Based on Lemma 23, $\bigoplus_{po}(\mathbf{R}) \neq id$. Therefore, based on (8), $\bigoplus_{po}(\mathbf{R}) = na$. \square

Proposition 8. Let $\Pi = \Pi_Q \cup \Pi_{po} \cup \Pi^P$ be a program obtained by merging Request transformation program Π_Q , permit-overrides combining algorithm transformation program Π_{po} and Policy \mathcal{P} transformation program with its components Π^P . Let M be an answer set of Π . Then,

$$\bigoplus_{po}(\mathbf{R}) = V \quad \text{if and only if} \quad algo(po, \mathcal{P}, V) \in M$$

where $\mathbf{R} = \langle \mathcal{R}_1(\mathcal{Q}), \dots, \mathcal{R}_n(\mathcal{Q}) \rangle$ be a sequence of policy value where each \mathcal{R}_i is a Rule in the sequence inside Policy \mathcal{P} .

Proof. It follows from Lemma 19, Lemma 20, Lemma 21, Lemma 22, Lemma 23 and Lemma 24 since the value of V only has six possibilities, i.e., $\{p, d, ip, id, idp, na\}$. \square

Combining Algorithm: First Applicable.

Proposition 9. Let $\Pi = \Pi_Q \cup \Pi_{fa} \cup \Pi^P$ be a program obtained by merging Request transformation program Π_Q , first-applicable combining algorithm transformation program Π_{fa} and Policy \mathcal{P} transformation program with its components Π^P . Let M be an answer set of Π . Then,

$$\bigoplus_{fa}(\mathbf{R}) = V \quad \text{if and only if} \quad algo(fa, \mathcal{P}, V) \in M$$

where $\mathbf{R} = \langle \llbracket \mathcal{R}_1 \rrbracket(\mathcal{Q}), \dots, \llbracket \mathcal{R}_n \rrbracket(\mathcal{Q}) \rangle$ be a sequence of policy value where each \mathcal{R}_i is a Rule in the sequence inside Policy \mathcal{P} .

Proof. (\Rightarrow) Suppose that $\bigoplus_{\text{fa}}(\mathbf{R}) = V$ holds. Then, as defined in (9), $\exists i : \llbracket \mathcal{R}_i \rrbracket(\mathcal{Q}) = V$ and $V \neq \text{na}$ and $\forall j : j < i \Rightarrow \llbracket \mathcal{R}_j \rrbracket(\mathcal{Q}) = \text{na}$. Based on Prop. 7, $\exists i : \text{val}(\mathcal{R}_i, V) \in M$ where $V \neq \text{na}$ and $\forall j : j < i \Rightarrow \text{val}(\mathcal{R}_j, \text{na}) \in M$. Based on the Policy transformation there is a rule $\text{decision_of}(\mathcal{P}, \mathcal{R}_i, V) \leftarrow \text{val}(\mathcal{R}_i, V)$ in Π and $\forall j : j < i$ we get that there are rules in the form $\text{decision_of}(\mathcal{P}, \mathcal{R}_j, \text{na}) \leftarrow \text{val}(\mathcal{R}_j, \text{na})$ in Π . Therefore, we have $\text{decision_of}(\mathcal{P}, \mathcal{R}_i, V) \in M$ and $\forall j : j < i$ we also have $\text{decision_of}(\mathcal{P}, \mathcal{R}_j, \text{na}) \in M$ since M is a minimal model for Π . Thus, by Lemma 1, $\text{algo}(\text{fa}, \mathcal{P}, V) \in M$.

(\Leftarrow) Suppose that $\text{algo}(\text{fa}, \mathcal{P}, V) \in M$. Based on Lemma 3, there is a clause in \mathcal{P} where $\text{algo}(\text{fa}, \mathcal{P}, V)$ as the head and the body is true under M . There are several rules in \mathcal{P} where $\text{algo}(\text{fa}, \mathcal{P}, V)$ as the head. We can see that in each rule the body contains $\exists i : \text{decision_of}(\mathcal{P}, \mathcal{R}_i, V), V \neq \text{na}$ and $\forall j : j < i$ the body also contains $\text{decision_of}(\mathcal{P}, \mathcal{R}_j, \text{na})$. Therefore $\exists i : \text{decision_of}(\mathcal{P}, \mathcal{R}_i, V) \in M$ and $\forall j : j < i, \text{decision_of}(\mathcal{P}, \mathcal{R}_j, \text{na}) \in M$. Based on Lemma 3, there is a clause where $\text{decision_of}(\mathcal{P}, \mathcal{R}_i, V)$ as the head and the body is true under M and $\forall j : j < i$, there is $\text{decision_of}(\mathcal{P}, \mathcal{R}_j, \text{na})$ as the head and the body is true under M . There is only one rule in \mathcal{P} where $\text{decision_of}(\mathcal{P}, \mathcal{R}_i, V)$ as the head, i.e., $\text{decision_of}(\mathcal{P}, \mathcal{R}_i, V) \leftarrow \text{val}(\mathcal{R}_i, V)$. The same case for $\text{decision_of}(\mathcal{P}, \mathcal{R}_j, \text{na})$. Then, $\exists i : M(\text{val}(\mathcal{R}_i, V)) = \top$ and $\forall j < i : M(\text{val}(\mathcal{R}_j, \text{na})) = \top$. Therefore, $\exists i : \text{val}(\mathcal{R}_i, V) \in M$ and $\forall j : j < i \Rightarrow \text{val}(\mathcal{R}_j, \text{na}) \in M$. Based on Prop. 7, $\exists i : \llbracket \mathcal{R}_i \rrbracket(\mathcal{Q}) = V$ and $\forall j : j < i \Rightarrow \llbracket \mathcal{R}_j \rrbracket(\mathcal{Q}) = \text{na}$ and \mathcal{R}_i and \mathcal{R}_j belong to the sequence inside Policy \mathcal{P} . Therefore, based on (8) we obtain $\bigoplus_{\text{fa}}(\mathbf{R}) = V$. \square

Combining Algorithm: Only-One-Applicable.

Lemma 25. Let $\Pi = \Pi_{\mathcal{Q}} \cup \Pi_{\text{ooa}} \cup \Pi^{\mathcal{P}}$ be a program obtained by merging Request transformation program $\Pi_{\mathcal{Q}}$, only-one-applicable combining algorithm transformation program Π_{ooa} and Policy \mathcal{P} transformation program with its components $\Pi^{\mathcal{P}}$. Let M be an answer set of Π . Then,

$$\bigoplus_{\text{ooa}}(\mathbf{R}) = i_{\text{dp}} \quad \text{if and only if} \quad \text{algo}(\text{ooa}, \mathcal{P}, i_{\text{dp}}) \in M$$

where $\mathbf{R} = \langle \mathcal{R}_1(\mathcal{Q}), \dots, \mathcal{R}_n(\mathcal{Q}) \rangle$ be a sequence of policy value where each \mathcal{R}_i is a Rule in the sequence inside Policy \mathcal{P} .

Proof. (\Rightarrow) Suppose that $\bigoplus_{\text{ooa}}(\mathbf{R}) = i_{\text{dp}}$ holds. Then, as defined in (10), we have that

1. $\exists i : \llbracket \mathcal{R}_i \rrbracket(\mathcal{Q}) = i_{\text{dp}}$.
Based on Prop. 7, $\text{val}(\mathcal{R}_i, i_{\text{dp}}) \in M$. Based on the Policy transformation, there is a rule $\text{decision_of}(\mathcal{P}, \mathcal{R}_i, i_{\text{dp}}) \leftarrow \text{val}(\mathcal{R}_i, i_{\text{dp}})$. Therefore, by Lemma 1, we find that $\text{decision_of}(\mathcal{P}, \mathcal{R}_i, i_{\text{dp}}) \in M$. Thus, by Lemma 1, $\text{algo}(\text{ooa}, \mathcal{P}, i_{\text{dp}}) \in M$.
2. $\exists i, j : \llbracket \mathcal{R}_i \rrbracket(\mathcal{Q}) = d$ and $\llbracket \mathcal{R}_j \rrbracket(\mathcal{Q}) = p$.
Based on Prop. 7, $\text{val}(\mathcal{R}_i, d) \in M$ and $\text{val}(\mathcal{R}_j, p) \in M$. Based on the Policy transformation, there are rules in Π with the form $\text{decision_of}(\mathcal{P}, \mathcal{R}_i, d) \leftarrow \text{val}(\mathcal{R}_i, d)$ and $\text{decision_of}(\mathcal{P}, \mathcal{R}_j, p) \leftarrow \text{val}(\mathcal{R}_j, p)$. Therefore, by Lemma 1, we find that $\text{decision_of}(\mathcal{P}, \mathcal{R}_i, d) \in M$ and $\text{decision_of}(\mathcal{P}, \mathcal{R}_j, p) \in M$. Thus, by Lemma 1, $\text{algo}(\text{ooa}, \mathcal{P}, i_{\text{dp}}) \in M$.

3. $\exists i, j : \llbracket \mathcal{R}_i \rrbracket(\mathcal{Q}) = i_d$ and $\llbracket \mathcal{R}_j \rrbracket(\mathcal{Q}) = p$.
Based on Prop. 7, $\text{val}(\mathcal{R}_i, i_d) \in M$ and $\text{val}(\mathcal{R}_j, p) \in M$. Based on the Policy transformation there are rules in Π in the form $\text{decision_of}(\mathcal{P}, \mathcal{R}_i, d) \leftarrow \text{val}(\mathcal{R}_i, i_d)$. and $\text{decision_of}(\mathcal{P}, \mathcal{R}_j, p) \leftarrow \text{val}(\mathcal{R}_j, p)$. Then, by Lemma 1, $\text{decision_of}(\mathcal{P}, \mathcal{R}_i, i_d) \in M$ and $\text{decision_of}(\mathcal{P}, \mathcal{R}_j, p) \in M$. Thus, by Lemma 1, $\text{algo}(\text{ooa}, \mathcal{P}, i_{dp}) \in M$.
4. $\exists i, j : \llbracket \mathcal{R}_i \rrbracket(\mathcal{Q}) = d$ and $\llbracket \mathcal{R}_j \rrbracket(\mathcal{Q}) = i_p$.
Based on Prop. 7, $\text{val}(\mathcal{R}_i, d) \in M$ and $\text{val}(\mathcal{R}_j, i_p) \in M$. Based on the Policy transformation, there are rules in Π in the form $\text{decision_of}(\mathcal{P}, \mathcal{R}_i, d) \leftarrow \text{val}(\mathcal{R}_i, d)$. and $\text{decision_of}(\mathcal{P}, \mathcal{R}_j, i_p) \leftarrow \text{val}(\mathcal{R}_j, i_p)$. Then, by Lemma 1, $\text{decision_of}(\mathcal{P}, \mathcal{R}_i, d) \in M$ and $\text{decision_of}(\mathcal{P}, \mathcal{R}_j, i_p) \in M$. Thus, by Lemma 1, $\text{algo}(\text{ooa}, \mathcal{P}, i_{dp}) \in M$.
5. $\exists i, j : \llbracket \mathcal{R}_i \rrbracket(\mathcal{Q}) = i_d$ and $\llbracket \mathcal{R}_j \rrbracket(\mathcal{Q}) = i_p$.
Based on Prop. 7, $\text{val}(\mathcal{R}_i, i_d) \in M$ and $\text{val}(\mathcal{R}_j, i_p) \in M$. Based on the Policy transformation there are rules in Π in the form $\text{decision_of}(\mathcal{P}, \mathcal{R}_i, i_d) \leftarrow \text{val}(\mathcal{R}_i, i_d)$. and $\text{decision_of}(\mathcal{P}, \mathcal{R}_j, i_p) \leftarrow \text{val}(\mathcal{R}_j, i_p)$. Then, by Lemma 1, $\text{decision_of}(\mathcal{P}, \mathcal{R}_i, i_d) \in M$ and $\text{decision_of}(\mathcal{P}, \mathcal{R}_j, i_p) \in M$. Thus by Lemma 1, $\text{algo}(\text{ooa}, \mathcal{P}, i_{dp}) \in M$.

(\Leftarrow) Suppose that $\text{algo}(\text{ooa}, \mathcal{P}, i_{dp}) \in M$. Based on Lemma 3, there is a rule where $\text{algo}(\text{ooa}, \mathcal{P}, i_{dp})$ as the head and the body is true under M . There are five rules in Π , i.e.,

1. $\text{algo}(\text{ooa}, \mathcal{P}, i_{dp}) \leftarrow \text{decision_of}(\mathcal{P}, \mathcal{R}, i_{dp})$.
Then, $M(\text{decision_of}(\mathcal{P}, \mathcal{R}, i_{dp})) = \top$. Therefore, $\text{decision_of}(\mathcal{P}, \mathcal{R}, i_{dp}) \in M$. Based on Lemma 3, there is a rule where $\text{decision_of}(\mathcal{P}, \mathcal{R}, i_{dp}) \in M$ as the head and the body is true under M . There is only one rule in Π , i.e., $\text{decision_of}(\mathcal{P}, \mathcal{R}, i_{dp}) \leftarrow \text{val}(\mathcal{R}, i_{dp})$. Then, $M(\text{val}(\mathcal{R}, i_{dp})) = \top$. Therefore, $\text{val}(\mathcal{R}, i_{dp}) \in M$. Based on Prop. 7, $\llbracket \mathcal{R} \rrbracket(\mathcal{Q}) = i_{dp}$ and \mathcal{R} belongs to the sequence inside Policy \mathcal{P} . Therefore, based on (10), $\bigoplus_{\text{ooa}}(\mathbf{R}) = i_{dp}$.
2. $\text{algo}(\text{ooa}, \mathcal{P}, i_{dp}) \leftarrow \text{decision_of}(\mathcal{P}, \mathcal{R}1, i_d), \text{decision_of}(\mathcal{P}, \mathcal{R}2, i_p)$.
Then, $M(\text{decision_of}(\mathcal{P}, \mathcal{R}1, i_d) \wedge \text{decision_of}(\mathcal{P}, \mathcal{R}2, i_p)) = \top$. Hence, we find that $\text{decision_of}(\mathcal{P}, \mathcal{R}1, i_d) \in M$ and $\text{decision_of}(\mathcal{P}, \mathcal{R}2, i_p) \in M$. Based on Lemma 3, there is a rule where $\text{decision_of}(\mathcal{P}, \mathcal{R}1, i_d) \in M$ as the head and the body is true under M , i.e., $\text{decision_of}(\mathcal{P}, \mathcal{R}1, i_d) \leftarrow \text{val}(\mathcal{R}1, i_d)$, and there is a rule where $\text{decision_of}(\mathcal{P}, \mathcal{R}2, i_p) \in M$ as the head and the body is true under M , i.e., $\text{decision_of}(\mathcal{P}, \mathcal{R}2, i_p) \leftarrow \text{val}(\mathcal{R}2, i_p)$. Therefore, $M(\text{val}(\mathcal{R}1, i_d)) = \top$ and $M(\text{val}(\mathcal{R}2, i_p)) = \top$. Then, $\text{val}(\mathcal{R}1, i_d) \in M$ and $\text{val}(\mathcal{R}2, i_p) \in M$. Based on Prop. 7, $\llbracket \mathcal{R}1 \rrbracket(\mathcal{Q}) = i_d$ and $\llbracket \mathcal{R}2 \rrbracket(\mathcal{Q}) = i_p$ and $\mathcal{R}1$ and $\mathcal{R}2$ belong to the sequence inside Policy \mathcal{P} . Therefore, based on (10), $\bigoplus_{\text{ooa}}(\mathbf{R}) = i_{dp}$.
3. $\text{algo}(\text{ooa}, \mathcal{P}, i_{dp}) \leftarrow \text{decision_of}(\mathcal{P}, \mathcal{R}1, i_d), \text{decision_of}(\mathcal{P}, \mathcal{R}2, p)$.
Then, we find that $M(\text{decision_of}(\mathcal{P}, \mathcal{R}1, i_d) \wedge \text{decision_of}(\mathcal{P}, \mathcal{R}2, p)) = \top$. Therefore, $\text{decision_of}(\mathcal{P}, \mathcal{R}1, i_d) \in M$ and $\text{decision_of}(\mathcal{P}, \mathcal{R}2, p) \in M$. Based on Lemma 3, there is a rule where $\text{decision_of}(\mathcal{P}, \mathcal{R}1, i_d) \in M$ as the head and the body is true under M , i.e., $\text{decision_of}(\mathcal{P}, \mathcal{R}1, i_d) \leftarrow \text{val}(\mathcal{R}1, i_d)$ and there is a rule where $\text{decision_of}(\mathcal{P}, \mathcal{R}2, p) \in M$ as the head and the body is true under M , i.e., $\text{decision_of}(\mathcal{P}, \mathcal{R}2, p) \leftarrow \text{val}(\mathcal{R}2, p)$. Then, we find that $M(\text{val}(\mathcal{R}1, i_d)) = \top$ and $M(\text{val}(\mathcal{R}2, p)) = \top$. Therefore, $\text{val}(\mathcal{R}1, i_d) \in M$ and $\text{val}(\mathcal{R}2, p) \in M$. Based on Prop. 7, $\llbracket \mathcal{R}1 \rrbracket(\mathcal{Q}) = i_d$ and $\llbracket \mathcal{R}2 \rrbracket(\mathcal{Q}) = p$ and $\mathcal{R}1$ and $\mathcal{R}2$ belong to the sequence inside Policy \mathcal{P} . Therefore, based on (10), $\bigoplus_{\text{ooa}}(\mathbf{R}) = i_{dp}$.
4. $\text{algo}(\text{ooa}, \mathcal{P}, i_{dp}) \leftarrow \text{decision_of}(\mathcal{P}, \mathcal{R}1, d), \text{decision_of}(\mathcal{P}, \mathcal{R}2, i_p)$.
Then we find that $M(\text{decision_of}(\mathcal{P}, \mathcal{R}1, d) \wedge \text{decision_of}(\mathcal{P}, \mathcal{R}2, i_p)) = \top$. Therefore, $\text{decision_of}(\mathcal{P}, \mathcal{R}1, d) \in M$ and $\text{decision_of}(\mathcal{P}, \mathcal{R}2, i_p) \in M$. Based on Lemma

- 3, there is a rule where $\text{decision_of}(\mathcal{P}, \mathcal{R}1, d) \in M$ as the head and the body is true under M , i.e., $\text{decision_of}(\mathcal{P}, \mathcal{R}1, d) \leftarrow \text{val}(\mathcal{R}1, d)$ and there is a rule where $\text{decision_of}(\mathcal{P}, \mathcal{R}2, i_p) \in M$ as the head and the body is true under M , i.e., $\text{decision_of}(\mathcal{P}, \mathcal{R}2, i_p) \leftarrow \text{val}(\mathcal{R}2, i_p)$. Then, we find that $M(\text{val}(\mathcal{R}1, d)) = \top$ and $M(\text{val}(\mathcal{R}2, i_p)) = \top$. Therefore, $\text{val}(\mathcal{R}1, d) \in M$ and $\text{val}(\mathcal{R}2, i_p) \in M$. Based on Prop. 7, $\llbracket \mathcal{R}1 \rrbracket(\mathcal{Q}) = d$ and $\llbracket \mathcal{R}2 \rrbracket(\mathcal{Q}) = i_p$ and $\mathcal{R}1$ and $\mathcal{R}2$ belong to the sequence inside Policy \mathcal{P} . Therefore, based on (10), $\bigoplus_{\text{ooa}}(\mathbf{R}) = i_{dp}$
5. $\text{algo}(\text{ooa}, \mathcal{P}, i_{dp}) \leftarrow \text{decision_of}(\mathcal{P}, \mathcal{R}1, d), \text{decision_of}(\mathcal{P}, \mathcal{R}2, p)$. Then we find that $M(\text{decision_of}(\mathcal{P}, \mathcal{R}1, d) \wedge \text{decision_of}(\mathcal{P}, \mathcal{R}2, p)) = \top$. Therefore, $\text{decision_of}(\mathcal{P}, \mathcal{R}1, d) \in M$ and $\text{decision_of}(\mathcal{P}, \mathcal{R}2, p) \in M$. Based on Lemma 3, there is a rule where $\text{decision_of}(\mathcal{P}, \mathcal{R}1, d) \in M$ as the head and the body is true under M , i.e., $\text{decision_of}(\mathcal{P}, \mathcal{R}1, d) \leftarrow \text{val}(\mathcal{R}1, d)$ and there is a rule where $\text{decision_of}(\mathcal{P}, \mathcal{R}2, p) \in M$ as the head and the body is true under M , i.e., $\text{decision_of}(\mathcal{P}, \mathcal{R}2, p) \leftarrow \text{val}(\mathcal{R}2, p)$. Then, we find that $M(\text{val}(\mathcal{R}1, d)) = \top$ and $M(\text{val}(\mathcal{R}2, p)) = \top$. Therefore, $\text{val}(\mathcal{R}1, d) \in M$ and $\text{val}(\mathcal{R}2, p) \in M$. Based on Prop. 7, $\llbracket \mathcal{R}1 \rrbracket(\mathcal{Q}) = d$ and $\llbracket \mathcal{R}2 \rrbracket(\mathcal{Q}) = p$ and $\mathcal{R}1$ and $\mathcal{R}2$ belong to the sequence inside Policy \mathcal{P} . Therefore, based on (10), $\bigoplus_{\text{ooa}}(\mathbf{R}) = i_{dp}$ \square

Lemma 26. Let $\Pi = \Pi_{\mathcal{Q}} \cup \Pi_{\text{ooa}} \cup \Pi^{\mathcal{P}}$ be a program obtained by merging Request transformation program $\Pi_{\mathcal{Q}}$, only-one-applicable combining algorithm transformation program Π_{ooa} and Policy \mathcal{P} transformation program with its components $\Pi^{\mathcal{P}}$. Let M be an answer set of Π . Then,

$$\bigoplus_{\text{ooa}}(\mathbf{R}) = i_d \quad \text{if and only if} \quad \text{algo}(\text{ooa}, \mathcal{P}, i_d) \in M$$

where $\mathbf{R} = \langle \mathcal{R}_1(\mathcal{Q}), \dots, \mathcal{R}_n(\mathcal{Q}) \rangle$ be a sequence of policy value where each \mathcal{R}_i is a Rule in the sequence inside Policy \mathcal{P} .

Proof. Suppose that $\bigoplus_{\text{ooa}}(\mathbf{R}) = i_d$ holds. Then, as defined in (10), we have that

1. $\forall i : \llbracket \mathcal{R}_i \rrbracket(\mathcal{Q}) \neq (p \text{ or } i_p \text{ or } i_{dp})$ and $\exists j : s_j = i_d$
Based on Prop. 7, $\forall i : \text{val}(\mathcal{R}_i, p) \notin M$, $\text{val}(\mathcal{R}_i, i_p) \notin M$ and $\text{val}(\mathcal{R}_i, i_{dp}) \notin M$. Based on Prop. 7, $\exists j : \text{val}(\mathcal{R}_j, i_d) \in M$. Based on Lemma 25, $\text{algo}(\text{ooa}, \mathcal{P}, i_{dp}) \notin M$ since if it is in M , there is a Rule \mathcal{R} such that $\llbracket \mathcal{R} \rrbracket(\mathcal{Q}) = (p \text{ or } i_p \text{ or } i_{dp})$ Based on the Policy transformation there is a rule $\text{decision_of}(\mathcal{P}, \mathcal{R}_j, i_d) \leftarrow \text{val}(\mathcal{R}_j, i_d)$. Then, by Lemma 1, $\text{decision_of}(\mathcal{P}, \mathcal{R}_j, i_d) \in M$. Then, by Lemma 1, we obtain $\text{algo}(\text{ooa}, \mathcal{P}, i_d) \in M$.
2. $\forall i : \llbracket \mathcal{R}_i \rrbracket(\mathcal{Q}) \neq (p \text{ or } i_p \text{ or } i_{dp})$ and $\exists j, k : j \neq k$ and $s_j = s_k = d$ Based on Prop. 7, $\forall i : \text{val}(\mathcal{R}_i, p) \notin M$, $\text{val}(\mathcal{R}_i, i_p) \notin M$ and $\text{val}(\mathcal{R}_i, i_{dp}) \notin M$ since if they are in M it will lead a contradiction. Based on Prop. 7, $\exists j, k : j \neq k$ and $\text{val}(\mathcal{R}_j, d) \in M$ and $\text{val}(\mathcal{R}_k, d) \in M$. Based on Lemma 25, $\text{algo}(\text{ooa}, \mathcal{P}, i_{dp}) \notin M$ since if it is in M , there is a Rule \mathcal{R} such that $\llbracket \mathcal{R} \rrbracket(\mathcal{Q}) = (p \text{ or } i_p \text{ or } i_{dp})$ Based on the Policy transformation there is a rule $\text{decision_of}(\mathcal{P}, \mathcal{R}_j, d) \leftarrow \text{val}(\mathcal{R}_j, d)$. and there is a rule $\text{decision_of}(\mathcal{P}, \mathcal{R}_k, d) \leftarrow \text{val}(\mathcal{R}_k, d)$. Therefore $\text{decision_of}(\mathcal{P}, \mathcal{R}_j, d) \in M$ and $\text{decision_of}(\mathcal{P}, \mathcal{R}_k, d) \in M$ since M is the minimal model of Π . Thus, $\text{algo}(\text{ooa}, \mathcal{P}, i_d) \in M$ since M is the minimal model of Π .

(\Leftarrow) Suppose that $\text{algo}(\text{ooa}, \mathcal{P}, i_d) \in M$. Based on Lemma 3, there is a rule where $\text{algo}(\text{ooa}, \mathcal{P}, i_d)$ as the head and the body is true under M . There are rules in Π where $\text{algo}(\text{ooa}, \mathcal{P}, i_d)$ as the head, i.e.,

1. $\text{algo}(\text{ooa}, \mathcal{P}, i_d) \leftarrow \text{not } \text{algo}(\text{ooa}, \mathcal{P}, i_{dp}), \text{decision_of}(\mathcal{P}, \mathcal{R}, i_d)$.
Then we find that $M(\text{not } \text{algo}(\text{ooa}, \mathcal{P}, i_{dp}) \wedge \text{decision_of}(\mathcal{P}, \mathcal{R}, i_d)) = \top$. Therefore, $\text{algo}(\text{ooa}, \mathcal{P}, i_{dp}) \notin M$ and $\text{decision_of}(\mathcal{P}, \mathcal{R}, i_d) \in M$. Based on Lemma 25, $\bigoplus_{\text{ooa}}(\mathbf{R}) \neq i_{dp}$. Based on Lemma 3, there is a rule where $\text{decision_of}(\mathcal{P}, \mathcal{R}, i_d)$ as the head and the body is true under M , i.e., $\text{decision_of}(\mathcal{P}, \mathcal{R}, i_d) \leftarrow \text{val}(\mathcal{R}, i_d)$. Then we find that $M(\text{val}(\mathcal{R}, i_d)) = \top$. Therefore, $\text{val}(\mathcal{R}, i_d) \in M$. Based on Prop. 7, $\llbracket \mathcal{R} \rrbracket(\mathcal{Q}) = i_d$. Based on (10), $\forall i : \llbracket \mathcal{R}_i \rrbracket(\mathcal{Q}) \neq (p \text{ or } i_p \text{ or } i_{dp})$ since $\bigoplus_{\text{ooa}}(\mathbf{R}) \neq i_{dp}$. Therefore, based on (10), $\bigoplus_{\text{ooa}}(\mathbf{R}) = i_d$.
2. $\text{algo}(\text{ooa}, \mathcal{P}, i_d) \leftarrow \text{not } \text{algo}(\text{ooa}, \mathcal{P}, i_{dp}), \text{decision_of}(\mathcal{P}, \mathcal{R}_1, d), \text{decision_of}(\mathcal{P}, \mathcal{R}_2, d), \mathcal{R}_1 \neq \mathcal{R}_2$.
Then, $M(\text{not } \text{algo}(\text{ooa}, \mathcal{P}, i_{dp}) \wedge \text{decision_of}(\mathcal{P}, \mathcal{R}_1, d) \wedge \text{decision_of}(\mathcal{P}, \mathcal{R}_2, d)) = \top$, $\mathcal{R}_1 \neq \mathcal{R}_2$. Therefore, $\text{algo}(\text{ooa}, \mathcal{P}, i_{dp}) \notin M$ and $\text{decision_of}(\mathcal{P}, \mathcal{R}_1, d) \in M$ and $\text{decision_of}(\mathcal{P}, \mathcal{R}_2, d) \in M$, $\mathcal{R}_1 \neq \mathcal{R}_2$. Based on Lemma 25, $\bigoplus_{\text{ooa}}(\mathbf{R}) \neq i_{dp}$. Based on Lemma 3, there is a rule where $\text{decision_of}(\mathcal{P}, \mathcal{R}_1, d)$ as the head and the body is true under M , i.e., $\text{decision_of}(\mathcal{P}, \mathcal{R}_1, d) \leftarrow \text{val}(\mathcal{R}_1, d)$. Then we find that $M(\text{val}(\mathcal{R}_1, d)) = \top$. Therefore, $\text{val}(\mathcal{R}_1, d) \in M$. Based on Prop. 7, $\llbracket \mathcal{R}_1 \rrbracket(\mathcal{Q}) = d$. Based on Lemma 3, there is a rule where $\text{decision_of}(\mathcal{P}, \mathcal{R}_2, d)$ as the head and the body is true under M , $\mathcal{R}_1 \neq \mathcal{R}_2$. There is only one rule in Π where $\text{decision_of}(\mathcal{P}, \mathcal{R}_2, d)$ as the head, i.e., $\text{decision_of}(\mathcal{P}, \mathcal{R}_2, d) \leftarrow \text{val}(\mathcal{R}_2, d)$, $\mathcal{R}_1 \neq \mathcal{R}_2$. Then we find that $M(\text{val}(\mathcal{R}_2, d)) = \top$, $\mathcal{R}_1 \neq \mathcal{R}_2$. Therefore, $\text{val}(\mathcal{R}_2, d) \in M$, $\mathcal{R}_1 \neq \mathcal{R}_2$. Based on Prop. 7, $\llbracket \mathcal{R}_2 \rrbracket(\mathcal{Q}) = d$. Based on (10), $\forall i : \llbracket \mathcal{R}_i \rrbracket(\mathcal{Q}) \neq (p \text{ or } i_p \text{ or } i_{dp})$ since $\bigoplus_{\text{ooa}}(\mathbf{R}) \neq i_{dp}$. Therefore, based on (10), $\bigoplus_{\text{ooa}}(\mathbf{R}) = i_d$. \square

Lemma 27. Let $\Pi = \Pi_{\mathcal{Q}} \cup \Pi_{\text{ooa}} \cup \Pi^{\mathcal{P}}$ be a program obtained by merging Request transformation program $\Pi_{\mathcal{Q}}$, only-one-applicable combining algorithm transformation program Π_{ooa} and Policy \mathcal{P} transformation program with its components $\Pi^{\mathcal{P}}$. Let M be an answer set of Π . Then,

$$\bigoplus_{\text{ooa}}(\mathbf{R}) = i_p \quad \text{if and only if} \quad \text{algo}(\text{ooa}, \mathcal{P}, i_p) \in M$$

where $\mathbf{R} = \langle \mathcal{R}_1(\mathcal{Q}), \dots, \mathcal{R}_n(\mathcal{Q}) \rangle$ be a sequence of policy value where each \mathcal{R}_i is a Rule in the sequence inside Policy \mathcal{P} .

Proof. Note: The proof is similar to the proof of Lemma 26.

Lemma 28. Let $\Pi = \Pi_{\mathcal{Q}} \cup \Pi_{\text{ooa}} \cup \Pi^{\mathcal{P}}$ be a program obtained by merging Request transformation program $\Pi_{\mathcal{Q}}$, only-one-applicable combining algorithm transformation program Π_{ooa} and Policy \mathcal{P} transformation program with its components $\Pi^{\mathcal{P}}$. Let M be an answer set of Π . Then,

$$\bigoplus_{\text{ooa}}(\mathbf{R}) = p \quad \text{if and only if} \quad \text{algo}(\text{ooa}, \mathcal{P}, p) \in M$$

where $\mathbf{R} = \langle \mathcal{R}_1(\mathcal{Q}), \dots, \mathcal{R}_n(\mathcal{Q}) \rangle$ be a sequence of policy value where each \mathcal{R}_i is a Rule in the sequence inside Policy \mathcal{P} .

Proof. (\Rightarrow) Suppose that $\bigoplus_{\text{ooa}}(\mathbf{R}) = p$ holds. Then, as defined in (10) we have that $\exists i : \llbracket \mathcal{R}_i \rrbracket(\mathcal{Q}) = d$ and $\forall j : j \neq i, \llbracket \mathcal{R}_j \rrbracket(\mathcal{Q}) = \text{na}$. Based on Prop. 7, $\exists i : \text{val}(\mathcal{R}_i, p) \in M$. Based on Lemma 25, $\text{algo}(\text{ooa}, \text{ooa}, i_{dp}) \notin M$ since if it is in M , there exists a Rule \mathcal{R} in Policy \mathcal{P} sequence such that $\llbracket \mathcal{R} \rrbracket(\mathcal{Q}) = i_{dp}$. Based on Lemma

26, $\text{algo}(\text{ooa}, \text{ooa}, i_d) \notin M$ since if it is in M , there exists a Rule \mathcal{R} in Policy \mathcal{P} sequence such that $\llbracket \mathcal{R} \rrbracket(\mathcal{Q}) = i_d$ or there are at least two Rule elements \mathcal{R}_1 and \mathcal{R}_2 , $\mathcal{R}_1 \neq \mathcal{R}_2$, such that $\llbracket \mathcal{R}_1 \rrbracket(\mathcal{Q}) = \llbracket \mathcal{R}_2 \rrbracket(\mathcal{Q}) = d$. Based on Lemma 27, $\text{algo}(\text{ooa}, \text{ooa}, i_p) \notin M$ since if it is in M , there exists a Rule \mathcal{R} in Policy \mathcal{P} sequence such that $\llbracket \mathcal{R} \rrbracket(\mathcal{Q}) = i_p$ or there are at least two Rule elements \mathcal{R}_1 and \mathcal{R}_2 , $\mathcal{R}_1 \neq \mathcal{R}_2$, such that $\llbracket \mathcal{R}_1 \rrbracket(\mathcal{Q}) = \llbracket \mathcal{R}_2 \rrbracket(\mathcal{Q}) = p$. Based on the Policy transformation there is a rule $\text{decision_of}(\mathcal{P}, \mathcal{R}_i, p) \leftarrow \text{val}(\mathcal{R}_i, p)$. Therefore $\text{decision_of}(\mathcal{P}, \mathcal{R}_i, p) \in M$ and $\text{decision_of}(\mathcal{P}, \mathcal{R}_j, na) \in M$ since M is the minimal model of Π . Thus, $\text{algo}(\text{ooa}, \mathcal{P}, p) \in M$ since M is the minimal model of Π .

(\Leftarrow) Suppose that $\text{algo}(\text{ooa}, \mathcal{P}, p) \in M$. Based on Lemma 3, there is a rule where $\text{algo}(\text{ooa}, \mathcal{P}, d)$ as the head and the body is true under M . There are rules in Π where $\text{algo}(\text{ooa}, \mathcal{P}, d)$ as the head, i.e., $\text{algo}(\text{ooa}, \mathcal{P}, i_d) \leftarrow \text{not algo}(\text{ooa}, \mathcal{P}, i_{dp}), \text{not algo}(\text{ooa}, \mathcal{P}, i_d), \text{not algo}(\text{ooa}, \mathcal{P}, i_p), \text{decision_of}(\mathcal{P}, \mathcal{R}, p)$.

Then, we find that $M(\text{not algo}(\text{ooa}, \mathcal{P}, i_{dp}) \wedge \text{not algo}(\text{ooa}, \mathcal{P}, i_d) \wedge \text{not algo}(\text{ooa}, \mathcal{P}, i_p) \wedge \text{decision_of}(\mathcal{P}, \mathcal{R}, i_d)) = \top$. Therefore, $\text{algo}(\text{ooa}, \mathcal{P}, i_{dp}) \notin M$, $\text{algo}(\text{ooa}, \mathcal{P}, i_d) \notin M$, $\text{algo}(\text{ooa}, \mathcal{P}, i_p) \notin M$ and $\text{decision_of}(\mathcal{P}, \mathcal{R}, i_d) \in M$. By Lemma 25, $\bigoplus_{\text{ooa}}(\mathbf{R}) \neq i_{dp}$. By Lemma 26, $\bigoplus_{\text{ooa}}(\mathbf{R}) \neq i_d$. By Lemma 27, $\bigoplus_{\text{ooa}}(\mathbf{R}) \neq i_p$. By Lemma 3, there is a rule where $\text{decision_of}(\mathcal{P}, \mathcal{R}, p)$ as the head and the body is true under M , i.e., $\text{decision_of}(\mathcal{P}, \mathcal{R}, p) \leftarrow \text{val}(\mathcal{R}, p)$. Then we find that $M(\text{val}(\mathcal{R}, i_d)) = \top$. Therefore, $\text{val}(\mathcal{R}, p) \in M$. Based on Prop. 7, $\llbracket \mathcal{R} \rrbracket(\mathcal{Q}) = p$. Based on (10), $\forall i : \llbracket \mathcal{R}_i \rrbracket(\mathcal{Q}) \neq (p \text{ or } i_p \text{ or } i_{dp})$ since $\bigoplus_{\text{ooa}}(\mathbf{R}) \neq i_{dp}$. Based on (10), $\forall i : \llbracket \mathcal{R}_i \rrbracket(\mathcal{Q}) \neq i_d$ or there are no two rules which $\llbracket \mathcal{R}_1 \rrbracket(\mathcal{Q}) = \llbracket \mathcal{R}_2 \rrbracket(\mathcal{Q}) = d$ since $\bigoplus_{\text{ooa}}(\mathbf{R}) \neq i_d$. Based on (10), $\forall i : \llbracket \mathcal{R}_i \rrbracket(\mathcal{Q}) \neq i_p$ or p since $\bigoplus_{\text{ooa}}(\mathbf{R}) \neq i_p$. Therefore, based on (10), $\bigoplus_{\text{ooa}}(\mathbf{R}) = p$. \square

Lemma 29. Let $\Pi = \Pi_{\mathcal{Q}} \cup \Pi_{\text{ooa}} \cup \Pi^{\mathcal{P}}$ be a program obtained by merging Request transformation program $\Pi_{\mathcal{Q}}$, only-one-applicable combining algorithm transformation program Π_{ooa} and Policy \mathcal{P} transformation program with its components $\Pi^{\mathcal{P}}$. Let M be an answer set of Π . Then,

$$\bigoplus_{\text{ooa}}(\mathbf{R}) = d \quad \text{if and only if} \quad \text{algo}(\text{ooa}, \mathcal{P}, d) \in M$$

where $\mathbf{R} = \langle \mathcal{R}_1(\mathcal{Q}), \dots, \mathcal{R}_n(\mathcal{Q}) \rangle$ be a sequence of policy value where each \mathcal{R}_i is a Rule in the sequence inside Policy \mathcal{P} .

Proof. Note: The proof is similar to the proof of Lemma 28.

Lemma 30. Let $\Pi = \Pi_{\mathcal{Q}} \cup \Pi_{\text{ooa}} \cup \Pi^{\mathcal{P}}$ be a program obtained by merging Request transformation program $\Pi_{\mathcal{Q}}$, only-one-applicable combining algorithm transformation program Π_{ooa} and Policy \mathcal{P} transformation program with its components $\Pi^{\mathcal{P}}$. Let M be an answer set of Π . Then,

$$\bigoplus_{\text{ooa}}(\mathbf{R}) = na \quad \text{if and only if} \quad \text{algo}(\text{ooa}, \mathcal{P}, na) \in M$$

where $\mathbf{R} = \langle \mathcal{R}_1(\mathcal{Q}), \dots, \mathcal{R}_n(\mathcal{Q}) \rangle$ be a sequence of policy value where each \mathcal{R}_i is a Rule in the sequence inside Policy \mathcal{P} .

Proof. (\Rightarrow) Suppose that $\bigoplus_{\text{ooa}}(\mathbf{R}) = na$ holds. Then, as defined in (8) we have that $\bigoplus_{\text{ooa}}(\mathbf{R}) \neq i_{dp}$, $\bigoplus_{\text{ooa}}(\mathbf{R}) \neq i_d$, $\bigoplus_{\text{ooa}}(\mathbf{R}) \neq i_p$, $\bigoplus_{\text{ooa}}(\mathbf{R}) \neq d$, and $\bigoplus_{\text{ooa}}(\mathbf{R}) \neq p$.

By Lemma 25, $\text{algo}(\text{ooa}, \mathcal{P}, i_{dp}) \notin M$. By Lemma 26, $\text{algo}(\text{ooa}, \mathcal{P}, i_d) \notin M$. By Lemma 27, $\text{algo}(\text{ooa}, \mathcal{P}, i_p) \notin M$. By Lemma 28, $\text{algo}(\text{ooa}, \mathcal{P}, p) \notin M$. By Lemma 29, $\text{algo}(\text{ooa}, \mathcal{P}, d) \notin M$. Thus, $M(\text{not } \text{algo}(\text{ooa}, \mathcal{P}, i_{dp}) \wedge \text{not } \text{algo}(\text{ooa}, \mathcal{P}, i_d) \wedge \text{not } \text{algo}(\text{ooa}, \mathcal{P}, i_p) \wedge \text{not } \text{algo}(\text{ooa}, \mathcal{P}, d) \wedge \text{not } \text{algo}(\text{ooa}, \mathcal{P}, p)) = \top$. Therefore, $\text{algo}(\text{ooa}, \mathcal{P}, na) \in M$ since M is the minimal model of Π .

(\Leftarrow) Suppose that $\text{algo}(\text{ooa}, \mathcal{P}, na) \in M$. Based on Lemma 3, there is a rule where $\text{algo}(\text{ooa}, \mathcal{P}, na)$ as the head and the body is true under M . There is only a rule in Π where $\text{algo}(\text{ooa}, \mathcal{P}, na)$ as the head, i.e., $\text{algo}(\text{ooa}, \mathcal{P}, na) \leftarrow \text{not } \text{algo}(\text{ooa}, \mathcal{P}, i_{dp}), \text{not } \text{algo}(\text{ooa}, \mathcal{P}, i_d), \text{not } \text{algo}(\text{ooa}, \mathcal{P}, i_p), \text{not } \text{algo}(\text{ooa}, \mathcal{P}, d), \text{not } \text{algo}(\text{ooa}, \mathcal{P}, p)$. Then we find that $M(\text{not } \text{algo}(\text{ooa}, \mathcal{P}, i_{dp}) \wedge \text{not } \text{algo}(\text{ooa}, \mathcal{P}, i_d) \wedge \text{not } \text{algo}(\text{ooa}, \mathcal{P}, i_p) \wedge \text{not } \text{algo}(\text{ooa}, \mathcal{P}, d) \wedge \text{not } \text{algo}(\text{ooa}, \mathcal{P}, p)) = \top$. Therefore, $\text{algo}(\text{ooa}, \mathcal{P}, i_{dp}) \notin M$, $\text{algo}(\text{ooa}, \mathcal{P}, i_d) \notin M$, $\text{algo}(\text{ooa}, \mathcal{P}, i_p) \notin M$, $\text{algo}(\text{ooa}, \mathcal{P}, d) \notin M$ and $\text{algo}(\text{ooa}, \mathcal{P}, p) \notin M$. By Lemma 19, $\bigoplus_{\text{ooa}}(\mathbf{R}) \neq i_{dp}$. By Lemma 20, $\bigoplus_{\text{ooa}}(\mathbf{R}) \neq i_d$. By Lemma 21, $\bigoplus_{\text{ooa}}(\mathbf{R}) \neq i_p$. By Lemma 22, $\bigoplus_{\text{ooa}}(\mathbf{R}) \neq p$. By Lemma 23, $\bigoplus_{\text{ooa}}(\mathbf{R}) \neq d$. Therefore, based on (8), $\bigoplus_{\text{ooa}}(\mathbf{R}) = na$ \square

Proposition 10. Let $\Pi = \Pi_{\mathcal{Q}} \cup \Pi_{\text{ooa}} \cup \Pi^{\mathcal{P}}$ be a program obtained by merging Request transformation program $\Pi_{\mathcal{Q}}$, only-one-applicable combining algorithm transformation program Π_{ooa} and Policy \mathcal{P} transformation program with its components $\Pi^{\mathcal{P}}$. Let M be an answer set of Π . Then,

$$\bigoplus_{\text{ooa}}(\mathbf{R}) = V \quad \text{if and only if} \quad \text{algo}(\text{ooa}, \mathcal{P}, V) \in M$$

where $\mathbf{R} = \langle \mathcal{R}_1(\mathcal{Q}), \dots, \mathcal{R}_n(\mathcal{Q}) \rangle$ be a sequence of policy value where each \mathcal{R}_i is a Rule in the sequence inside Policy \mathcal{P} .

Proof. It follows from Lemma 25, Lemma 26, Lemma 27, Lemma 28, Lemma 29 and Lemma 30 since the value of V only has six possibilities, i.e., $\{p, d, i_p, i_d, i_{dp}, na\}$. \square

Evaluation to Combining Algorithms.

Proposition 11. Let $\Pi = \Pi_{\mathcal{Q}} \cup \Pi_{\text{ComblD}} \cup \Pi^{\mathcal{P}}$ be a program obtained by merging Request transformation program $\Pi_{\mathcal{Q}}$, combining algorithm transformation program Π_{ComblD} and Policy \mathcal{P} transformation program with its components $\Pi^{\mathcal{P}}$. Let M be an answer set of Π . Then,

$$\bigoplus_{\text{ComblD}}(\mathbf{R}) = V \quad \text{if and only if} \quad \text{algo}(\text{ComblD}, \mathcal{P}, V) \in M$$

where $\mathbf{R} = \langle \mathcal{R}_1(\mathcal{Q}), \dots, \mathcal{R}_n(\mathcal{Q}) \rangle$ be a sequence of policy value where each \mathcal{R}_i is a Rule in the sequence inside Policy \mathcal{P} .

Proof. It follows from Prop. 8, Prop. 9 and Prop. 10. \square

Policy Evaluation.

Lemma 31. Let $\Pi = \Pi_{\mathcal{Q}} \cup \Pi^{\mathcal{P}}$ be a program obtained by merging Request transformation program $\Pi_{\mathcal{Q}}$ Policy \mathcal{P} transformation program and its components $\Pi^{\mathcal{P}}$. Let M be an answer set of Π . Then,

$$\llbracket \mathcal{P} \rrbracket(\mathcal{Q}) = i_d \text{ iff } \text{val}(\mathcal{P}, i_d) \in M.$$

Proof. (\Rightarrow) Suppose that $\llbracket \mathcal{P} \rrbracket(\mathcal{Q}) = i_d$ holds. Then, as defined in (7) we have that

1. $\llbracket \mathcal{T} \rrbracket(\mathcal{Q}) = \text{idt}$ and $\bigoplus_{\text{ComblD}}(\mathbf{R}) = d$. Based on Prop. 5 and Prop. 11, $\text{val}(\mathcal{T}, \text{idt}) \in M$ and $\text{algo}(\text{ComblD}, \mathcal{P}, d) \in M$. Thus, $M(\text{val}(\mathcal{T}, \text{idt}) \wedge \text{algo}(\text{ComblD}, \mathcal{P}, d)) = \top$. Hence, by Lemma 1, $\text{val}(\mathcal{P}, i_d) \in M$.
2. $\llbracket \mathcal{T} \rrbracket(\mathcal{Q}) = \text{idt}$ and $\bigoplus_{\text{ComblD}}(\mathbf{R}) = i_d$ and $\forall i : \llbracket \mathcal{R}_i \rrbracket(\mathcal{Q}) \neq \text{na}$. Based on Prop. 5 and Prop. 11, $\text{val}(\mathcal{T}, \text{idt}) \in M$ and $\text{algo}(\text{ComblD}, \mathcal{P}, i_d) \in M$. Based on Prop. 7, $\exists i : \text{val}(\mathcal{R}_i, V) \in M, V \neq \text{na}$. Therefore, we have $\text{decision_of}(\mathcal{P}, \mathcal{R}_i, V) \in M$ since M is the minimal model of Π . Thus, $M(\text{val}(\mathcal{T}, \text{idt}) \wedge \text{algo}(\text{ComblD}, \mathcal{P}, i_d) \wedge \text{decision_of}(\mathcal{P}, \mathcal{R}_i, V) \wedge (V \neq \text{na})) = \top$. Hence, by Lemma 1, $\text{val}(\mathcal{P}, i_d) \in M$.
3. $\llbracket \mathcal{T} \rrbracket(\mathcal{Q}) = m$ and $\bigoplus_{\text{ComblD}}(\mathbf{R}) = i_d$ and $\forall i : \llbracket \mathcal{R}_i \rrbracket(\mathcal{Q}) \neq \text{na}$. Based on Prop. 5 and Prop. 11, $\text{val}(\mathcal{T}, m) \in M$ and $\text{algo}(\text{ComblD}, \mathcal{P}, i_d) \in M$. Based on Prop. 7, $\exists i : \text{val}(\mathcal{R}_i, V) \in M, V \neq \text{na}$. Therefore, we have $\text{decision_of}(\mathcal{P}, \mathcal{R}_i, V) \in M$ since M is the minimal model of Π . Thus, $M(\text{val}(\mathcal{T}, m) \wedge \text{algo}(\text{ComblD}, \mathcal{P}, i_d) \wedge \text{decision_of}(\mathcal{P}, \mathcal{R}_i, V) \wedge (V \neq \text{na})) = \top$. Therefore, $\text{val}(\mathcal{P}, i_d) \in M$ since M is the minimal model of Π .

(\Leftarrow) Suppose that $\text{val}(\mathcal{P}, i_d) \in M$. Based on Lemma 3, there is a clause where $\text{val}(\mathcal{P}, i_d)$ as the head and the body is true under M . There are rules in Π where $\text{val}(\mathcal{P}, i_d)$ as the head, i.e.,

1. $\text{val}(\mathcal{P}, i_d) \leftarrow \text{val}(\mathcal{T}, \text{idt}), \text{algo}(\text{ComblD}, \mathcal{P}, d)$.
Then, $M(\text{val}(\mathcal{T}, \text{idt}) \wedge \text{algo}(\text{ComblD}, \mathcal{P}, d)) = \top$. Therefore, $\text{val}(\mathcal{T}, \text{idt}) \in M$ and $\text{algo}(\text{ComblD}, \mathcal{P}, d) \in M$. Based on Prop. 5 and Prop. 11, $\llbracket \mathcal{T} \rrbracket(\mathcal{Q}) = \text{idt}$ and $\bigoplus_{\text{ComblD}}(\mathbf{R}) = d$. Therefore, based on (7), $\llbracket \mathcal{P} \rrbracket(\mathcal{Q}) = i_d$.
2. $\text{val}(\mathcal{P}, i_d) \leftarrow \text{val}(\mathcal{T}, \text{idt}), \text{decision_of}(\mathcal{P}, \mathcal{R}, V), V \neq \text{na}, \text{algo}(\text{ComblD}, \mathcal{P}, i_d), i_d \neq d$.
Then, $M(\text{val}(\mathcal{T}, \text{idt}) \wedge \text{decision_of}(\mathcal{P}, \mathcal{R}, V) \wedge (V \neq \text{na}) \wedge \text{algo}(\text{ComblD}, \mathcal{P}, i_d) \wedge (i_d \neq d)) = \top$. Therefore, $\text{val}(\mathcal{T}, \text{idt}) \in M$, $\text{decision_of}(\mathcal{P}, \mathcal{R}, V) \in M$, $V \neq \text{na}$ and $\text{algo}(\text{ComblD}, \mathcal{P}, i_d) \in M$. Based on Prop. 5 and Prop. 11, $\llbracket \mathcal{T} \rrbracket(\mathcal{Q}) = \text{idt}$ and $\bigoplus_{\text{ComblD}}(\mathbf{R}) = i_d$. Based on Lemma 3, there is a clause where $\text{decision_of}(\mathcal{P}, \mathcal{R}, V)$ as the head and the body is true under M . There is a rule Π where $\text{decision_of}(\mathcal{P}, \mathcal{R}, V)$ as the head, i.e., $\text{decision_of}(\mathcal{P}, \mathcal{R}, V) \leftarrow \text{val}(\mathcal{R}, V)$. Then we find that $M(\text{val}(\mathcal{R}, V)) = \top$. Thus, $\text{val}(\mathcal{R}, V) \in M$. Based on Prop. 7, $\llbracket \mathcal{R} \rrbracket(\mathcal{Q}) = V, V \neq \text{na}$. Therefore, based on (7), $\llbracket \mathcal{P} \rrbracket(\mathcal{Q}) = i_d$.
3. $\text{val}(\mathcal{P}, i_d) \leftarrow \text{val}(\mathcal{T}, m), \text{decision_of}(\mathcal{P}, \mathcal{R}, V), V \neq \text{na}, \text{algo}(\text{ComblD}, \mathcal{P}, i_d)$.
Then we find that $M(\text{val}(\mathcal{T}, m) \wedge \text{decision_of}(\mathcal{P}, \mathcal{R}, V) \wedge (V \neq \text{na}) \wedge \text{algo}(\text{ComblD}, \mathcal{P}, i_d)) = \top$. Therefore, $\text{val}(\mathcal{T}, m) \in M$, $\text{decision_of}(\mathcal{P}, \mathcal{R}, V) \in M$, $V \neq \text{na}$ and $\text{algo}(\text{ComblD}, \mathcal{P}, i_d) \in M$. Based on Prop. 5 and Prop. 11, $\llbracket \mathcal{T} \rrbracket(\mathcal{Q}) = m$ and $\bigoplus_{\text{ComblD}}(\mathbf{R}) = i_d$. Based on Lemma 3, there is a clause where $\text{decision_of}(\mathcal{P}, \mathcal{R}, V)$ as the head and the body is true under M . There is a rule Π where $\text{decision_of}(\mathcal{P}, \mathcal{R}, V)$ as the head, i.e., $\text{decision_of}(\mathcal{P}, \mathcal{R}, V) \leftarrow \text{val}(\mathcal{R}, V)$. Then we find that $M(\text{val}(\mathcal{R}, V)) = \top$. Thus, $\text{val}(\mathcal{R}, V) \in M$. Based on Prop. 7, $\llbracket \mathcal{R} \rrbracket(\mathcal{Q}) = V, V \neq \text{na}$. Therefore, based on (7), $\llbracket \mathcal{P} \rrbracket(\mathcal{Q}) = i_d$. \square

Lemma 32. Let $\Pi = \Pi_{\mathcal{Q}} \cup \Pi^{\mathcal{P}}$ be a program obtained by merging Request transformation program $\Pi_{\mathcal{Q}}$ Policy \mathcal{P} transformation program and its components $\Pi^{\mathcal{P}}$. Let M be an answer set of Π . Then,

$$\llbracket \mathcal{P} \rrbracket(\mathcal{Q}) = i_p \quad \text{if and only if} \quad \text{val}(\mathcal{P}, i_p) \in M .$$

Proof. Note: The proof is similar with the proof in Lemma 31.

Lemma 33. Let $\Pi = \Pi_Q \cup \Pi^P$ be a program obtained by merging Request transformation program Π_Q Policy \mathcal{P} transformation program and its components Π^P . Let M be an answer set of Π . Then,

$$\llbracket \mathcal{P} \rrbracket(Q) = \text{na} \quad \text{if and only if} \quad \text{val}(\mathcal{P}, \text{na}) \in M.$$

Proof. (\Rightarrow) Suppose that $\llbracket \mathcal{P} \rrbracket(Q) = \text{na}$ holds. Then, as defined in (7) we have that

1. $\llbracket \mathcal{T} \rrbracket(Q) = \text{nm}$. Based on Prop. 5, $\text{val}(\mathcal{T}, \text{nm}) \in M$. Thus, $M(\text{val}(\mathcal{T}, \text{nm})) = \top$. Therefore, $\text{val}(\mathcal{P}, \text{na}) \in M$ since M is the minimal model of Π .
2. $\forall i : \llbracket \mathcal{R}_i \rrbracket(Q) = \text{na}$. Based on Prop. 7, $\forall i : \text{val}(\mathcal{R}_i, \text{na}) \in M$. Thus, $M(\text{val}(\mathcal{R}_1, \text{na}) \wedge \dots \wedge \text{val}(\mathcal{R}_n, \text{na})) = \top$. Therefore, $\text{val}(\mathcal{P}, \text{na}) \in M$ since M is the minimal model of Π .

(\Leftarrow) Suppose that $\text{val}(\mathcal{P}, i_d) \in M$. Based on Lemma 3, there is a clause where $\text{val}(\mathcal{P}, i_p)$ as the head and the body is true under M . There are rules in Π where $\text{val}(\mathcal{P}, i_p)$ as the head, i.e.,

1. $\text{val}(\mathcal{P}, \text{na}) \leftarrow \text{val}(\mathcal{T}, \text{nm})$. Then we find that $M(\text{val}(\mathcal{T}, \text{nm})) = \top$. Therefore, $\text{val}(\mathcal{T}, \text{nm}) \in M$. Based on Prop. 5, $\llbracket \mathcal{T} \rrbracket(Q) = \text{na}$. Therefore, based on (7), $\llbracket \mathcal{P} \rrbracket(Q) = \text{na}$.
2. $\text{val}(\mathcal{P}, \text{na}) \leftarrow \text{val}(\mathcal{R}_1, \text{na}), \dots, \text{val}(\mathcal{R}_n, \text{na})$. Then we find that $M(\text{val}(\mathcal{R}_1, \text{na}) \wedge \dots \wedge \text{val}(\mathcal{R}_n, \text{na})) = \top$. Therefore, $\forall i : \text{val}(\mathcal{R}_i, \text{na}) \in M$. Based on Prop. 7, $\forall i : \llbracket \mathcal{R}_i \rrbracket(Q) = \text{na}$. Therefore, based on (7), $\llbracket \mathcal{P} \rrbracket(Q) = \text{na}$. \square

Lemma 34. Let $\Pi = \Pi_Q \cup \Pi^P$ be a program obtained by merging Request transformation program Π_Q Policy \mathcal{P} transformation program and its components Π^P . Let M be an answer set of Π . Then,

$$\llbracket \mathcal{P} \rrbracket(Q) = i_{dp} \quad \text{if and only if} \quad \text{val}(\mathcal{P}, i_{dp}) \in M.$$

Proof. (\Rightarrow) Suppose that $\llbracket \mathcal{P} \rrbracket(Q) = i_{dp}$ holds. Then, as defined in (7) we have that

1. $\llbracket \mathcal{T} \rrbracket(Q) = \text{idt}$ and $\bigoplus_{\text{CombID}}(\mathbf{R}) = i_{dp}$ and $\forall i : \llbracket \mathcal{R}_i \rrbracket(Q) \neq \text{na}$. Based on Prop. 5 and Prop. 11, $\text{val}(\mathcal{T}, \text{idt}) \in M$ and $\text{algo}(\text{CombID}, \mathcal{P}, i_{dp}) \in M$. Based on Prop. 7, $\exists i : \text{val}(\mathcal{R}_i, V) \in M, V \neq \text{na}$. Therefore, we have $\text{decision_of}(\mathcal{P}, \mathcal{R}_i, V) \in M$ since M is the minimal model of Π . Thus, $M(\text{val}(\mathcal{T}, \text{idt}) \wedge \text{algo}(\text{CombID}, \mathcal{P}, i_{dp}) \wedge \text{decision_of}(\mathcal{P}, \mathcal{R}_i, V) \wedge (V \neq \text{na})) = \top$. Therefore, $\text{val}(\mathcal{P}, i_{dp}) \in M$ since M is the minimal model of Π .
2. $\llbracket \mathcal{T} \rrbracket(Q) = m$ and $\bigoplus_{\text{CombID}}(\mathbf{R}) = i_{dp}$ and $\forall i : \llbracket \mathcal{R}_i \rrbracket(Q) \neq \text{na}$. Based on Prop. 5 and Prop. 11, $\text{val}(\mathcal{T}, m) \in M$ and $\text{algo}(\text{CombID}, \mathcal{P}, i_{dp}) \in M$. Based on Prop. 7, $\exists i : \text{val}(\mathcal{R}_i, V) \in M, V \neq \text{na}$. Therefore, we have $\text{decision_of}(\mathcal{P}, \mathcal{R}_i, V) \in M$ since M is the minimal model of Π . Thus, $M(\text{val}(\mathcal{T}, m) \wedge \text{algo}(\text{CombID}, \mathcal{P}, i_{dp}) \wedge \text{decision_of}(\mathcal{P}, \mathcal{R}_i, V) \wedge (V \neq \text{na})) = \top$. Therefore, $\text{val}(\mathcal{P}, i_{dp}) \in M$ since M is the minimal model of Π .

(\Leftarrow) Suppose that $\text{val}(\mathcal{P}, i_{dp}) \in M$. Based on Lemma 3, there is a clause where $\text{val}(\mathcal{P}, i_{dp})$ as the head and the body is true under M . There are rules in Π where $\text{val}(\mathcal{P}, i_{dp})$ as the head, i.e.,

1. $\text{val}(\mathcal{P}, i_{dp}) \leftarrow \text{val}(\mathcal{T}, \text{idt}), \text{decision_of}(\mathcal{P}, \mathcal{R}, V), V \neq \text{na}, \text{algo}(\text{ComblD}, \mathcal{P}, i_{dp}), i_{dp} \neq d$. Then we find that $M(\text{val}(\mathcal{T}, \text{idt}) \wedge \text{decision_of}(\mathcal{P}, \mathcal{R}, V) \wedge (V \neq \text{na}) \wedge \text{algo}(\text{ComblD}, \mathcal{P}, i_{dp}) \wedge (i_{dp} \neq d)) = \top$. Therefore, $\text{val}(\mathcal{T}, \text{idt}) \in M$, $\text{decision_of}(\mathcal{P}, \mathcal{R}, V) \in M$, $V \neq \text{na}$ and $\text{algo}(\text{ComblD}, \mathcal{P}, i_{dp}) \in M$. Based on Prop. 5 and Prop. 11, $\llbracket \mathcal{T} \rrbracket(\mathcal{Q}) = \text{idt}$ and $\bigoplus_{\text{ComblD}}(\mathbf{R}) = i_{dp}$. Based on Lemma 3, there is a clause where $\text{decision_of}(\mathcal{P}, \mathcal{R}, V)$ as the head and the body is true under M . There is a rule Π where $\text{decision_of}(\mathcal{P}, \mathcal{R}, V)$ as the head, i.e., $\text{decision_of}(\mathcal{P}, \mathcal{R}, V) \leftarrow \text{val}(\mathcal{R}, V)$. Then we find that $M(\text{val}(\mathcal{R}, V)) = \top$. Thus, $\text{val}(\mathcal{R}, V) \in M$. Based on Prop. 7, $\llbracket \mathcal{R} \rrbracket(\mathcal{Q}) = V, V \neq \text{na}$. Therefore, based on (7), $\llbracket \mathcal{P} \rrbracket(\mathcal{Q}) = i_{dp}$.
2. $\text{val}(\mathcal{P}, i_{dp}) \leftarrow \text{val}(\mathcal{T}, m), \text{decision_of}(\mathcal{P}, \mathcal{R}, V), V \neq \text{na}, \text{algo}(\text{ComblD}, \mathcal{P}, i_{dp})$. Then we find that $M(\text{val}(\mathcal{T}, m) \wedge \text{decision_of}(\mathcal{P}, \mathcal{R}, V) \wedge (V \neq \text{na}) \wedge \text{algo}(\text{ComblD}, \mathcal{P}, i_{dp})) = \top$. Therefore, $\text{val}(\mathcal{T}, m) \in M$, $\text{decision_of}(\mathcal{P}, \mathcal{R}, V) \in M$, $V \neq \text{na}$ and $\text{algo}(\text{ComblD}, \mathcal{P}, i_{dp}) \in M$. Based on Prop. 5 and Prop. 11, $\llbracket \mathcal{T} \rrbracket(\mathcal{Q}) = m$ and $\bigoplus_{\text{ComblD}}(\mathbf{R}) = i_{dp}$. Based on Lemma 3, there is a clause where $\text{decision_of}(\mathcal{P}, \mathcal{R}, V)$ as the head and the body is true under M . There is a rule Π where $\text{decision_of}(\mathcal{P}, \mathcal{R}, V)$ as the head, i.e., $\text{decision_of}(\mathcal{P}, \mathcal{R}, V) \leftarrow \text{val}(\mathcal{R}, V)$. Then we find that $M(\text{val}(\mathcal{R}, V)) = \top$. Thus, $\text{val}(\mathcal{R}, V) \in M$. Based on Prop. 7, $\llbracket \mathcal{R} \rrbracket(\mathcal{Q}) = V, V \neq \text{na}$. Therefore, based on (7), $\llbracket \mathcal{P} \rrbracket(\mathcal{Q}) = i_{dp}$. \square

Lemma 35. Let $\Pi = \Pi_{\mathcal{Q}} \cup \Pi^{\mathcal{P}}$ be a program obtained by merging Request transformation program $\Pi_{\mathcal{Q}}$ Policy \mathcal{P} transformation program and its components $\Pi^{\mathcal{P}}$. Let M be an answer set of Π . Then,

$$\llbracket \mathcal{P} \rrbracket(\mathcal{Q}) = p \quad \text{if and only if} \quad \text{val}(\mathcal{P}, p) \in M.$$

Proof. (\Rightarrow) Suppose that $\llbracket \mathcal{P} \rrbracket(\mathcal{Q}) = p$ holds. Then, as defined in (7) we have that $\llbracket \mathcal{T} \rrbracket(\mathcal{Q}) = m$ and $\bigoplus_{\text{ComblD}}(\mathbf{R}) = p$ and $\forall i : \llbracket \mathcal{R}_i \rrbracket(\mathcal{Q}) \neq \text{na}$. Based on Prop. 5 and Prop. 11, $\text{val}(\mathcal{T}, m) \in M$ and $\text{algo}(\text{ComblD}, \mathcal{P}, p) \in M$. Based on Prop. 7, $\exists i : \text{val}(\mathcal{R}_i, V) \in M, V \neq \text{na}$. Therefore, we have $\text{decision_of}(\mathcal{P}, \mathcal{R}_i, V) \in M$ since M is the minimal model of Π . Thus, $M(\text{val}(\mathcal{T}, m) \wedge \text{algo}(\text{ComblD}, \mathcal{P}, p) \wedge \text{decision_of}(\mathcal{P}, \mathcal{R}_i, V) \wedge (V \neq \text{na})) = \top$. Therefore, $\text{val}(\mathcal{P}, p) \in M$ since M is the minimal model of Π .

(\Leftarrow) Suppose that $\text{val}(\mathcal{P}, i_d) \in M$. Based on Lemma 3, there is a clause where $\text{val}(\mathcal{P}, i_d)$ as the head and the body is true under M . There are rules in Π where $\text{val}(\mathcal{P}, i_d)$ as the head, i.e., $\text{val}(\mathcal{P}, p) \leftarrow \text{val}(\mathcal{T}, m), \text{decision_of}(\mathcal{P}, \mathcal{R}, V), V \neq \text{na}, \text{algo}(\text{ComblD}, \mathcal{P}, p)$. Then we find that $M(\text{val}(\mathcal{T}, m) \wedge \text{decision_of}(\mathcal{P}, \mathcal{R}, V) \wedge (V \neq \text{na}) \wedge \text{algo}(\text{ComblD}, \mathcal{P}, p)) = \top$. Therefore, $\text{val}(\mathcal{T}, m) \in M$, $\text{decision_of}(\mathcal{P}, \mathcal{R}, V) \in M$, $V \neq \text{na}$ and $\text{algo}(\text{ComblD}, \mathcal{P}, p) \in M$. Based on Prop. 5 and Prop. 11, $\llbracket \mathcal{T} \rrbracket(\mathcal{Q}) = m$ and $\bigoplus_{\text{ComblD}}(\mathbf{R}) = p$. Based on Lemma 3, there is a clause where $\text{decision_of}(\mathcal{P}, \mathcal{R}, V)$ as the head and the body is true under M . There is a rule Π where $\text{decision_of}(\mathcal{P}, \mathcal{R}, V)$ as the head, i.e., $\text{decision_of}(\mathcal{P}, \mathcal{R}, V) \leftarrow \text{val}(\mathcal{R}, V)$. Then we find that $M(\text{val}(\mathcal{R}, V)) = \top$. Thus, $\text{val}(\mathcal{R}, V) \in M$. Based on Prop. 7, $\llbracket \mathcal{R} \rrbracket(\mathcal{Q}) = V, V \neq \text{na}$. Therefore, based on (7), $\llbracket \mathcal{P} \rrbracket(\mathcal{Q}) = p$. \square

Lemma 36. Let $\Pi = \Pi_{\mathcal{Q}} \cup \Pi^{\mathcal{P}}$ be a program obtained by merging Request transformation program $\Pi_{\mathcal{Q}}$ Policy \mathcal{P} transformation program and its components $\Pi^{\mathcal{P}}$. Let M be an answer set of Π . Then,

$$\llbracket \mathcal{P} \rrbracket(\mathcal{Q}) = d \quad \text{if and only if} \quad \text{val}(\mathcal{P}, d) \in M.$$

Proof. (\Rightarrow) Suppose that $\llbracket \mathcal{P} \rrbracket(\mathcal{Q}) = d$ holds. Then, as defined in (7) we have that $\llbracket \mathcal{T} \rrbracket(\mathcal{Q}) = m$ and $\bigoplus_{\text{ComblD}}(\mathbf{R}) = d$ and $\forall i : \llbracket \mathcal{R}_i \rrbracket(\mathcal{Q}) \neq \text{na}$. Based on Prop. 5

and Prop. 11, $\text{val}(\mathcal{T}, m) \in M$ and $\text{algo}(\text{ComblD}, \mathcal{P}, d) \in M$. Based on Prop. 7, $\exists i : \text{val}(\mathcal{R}_i, V) \in M, V \neq \text{na}$. Therefore, we have $\text{decision_of}(\mathcal{P}, \mathcal{R}_i, V) \in M$ since M is the minimal model of Π . Thus, $M(\text{val}(\mathcal{T}, m) \wedge \text{algo}(\text{ComblD}, \mathcal{P}, d) \wedge \text{decision_of}(\mathcal{P}, \mathcal{R}_i, V) \wedge (V \neq \text{na})) = \top$. Therefore, $\text{val}(\mathcal{P}, d) \in M$ since M is the minimal model of Π .

(\Leftarrow) Suppose that $\text{val}(\mathcal{P}, i_d) \in M$. Based on Lemma 3, there is a clause where $\text{val}(\mathcal{P}, i_d)$ as the head and the body is true under M . There are rules in Π where $\text{val}(\mathcal{P}, i_d)$ as the head, i.e., $\text{val}(\mathcal{P}, d) \leftarrow \text{val}(\mathcal{T}, m), \text{decision_of}(\mathcal{P}, \mathcal{R}, V), V \neq \text{na}, \text{algo}(\text{ComblD}, \mathcal{P}, d)$. Then we find that $M(\text{val}(\mathcal{T}, m) \wedge \text{decision_of}(\mathcal{P}, \mathcal{R}, V) \wedge (V \neq \text{na}) \wedge \text{algo}(\text{ComblD}, \mathcal{P}, d)) = \top$. Therefore, $\text{val}(\mathcal{T}, m) \in M, \text{decision_of}(\mathcal{P}, \mathcal{R}, V) \in M, V \neq \text{na}$ and $\text{algo}(\text{ComblD}, \mathcal{P}, d) \in M$. Based on Prop. 5 and Prop. 11, $\llbracket \mathcal{T} \rrbracket(\mathcal{Q}) = m$ and $\oplus_{\text{ComblD}}(\mathbf{R}) = d$. Based on Lemma 3, there is a clause where $\text{decision_of}(\mathcal{P}, \mathcal{R}, V)$ as the head and the body is true under M . There is a rule Π where $\text{decision_of}(\mathcal{P}, \mathcal{R}, V)$ as the head, i.e., $\text{decision_of}(\mathcal{P}, \mathcal{R}, V) \leftarrow \text{val}(\mathcal{R}, V)$. Then we find that $M(\text{val}(\mathcal{R}, V)) = \top$. Thus, $\text{val}(\mathcal{R}, V) \in M$. Based on Prop. 7, $\llbracket \mathcal{R} \rrbracket(\mathcal{Q}) = V, V \neq \text{na}$. Therefore, based on (7), $\llbracket \mathcal{P} \rrbracket(\mathcal{Q}) = d$. \square

Proposition 12. *Let $\Pi = \Pi_{\mathcal{Q}} \cup \Pi^{\mathcal{P}}$ be a program obtained by merging Request transformation program $\Pi_{\mathcal{Q}}$ Policy \mathcal{P} transformation program and its components $\Pi^{\mathcal{P}}$. Let M be an answer set of Π . Then,*

$$\llbracket \mathcal{P} \rrbracket(\mathcal{Q}) = V \quad \text{if and only if} \quad \text{val}(\mathcal{P}, V) \in M.$$

Proof. It follows from Lemma 31, Lemma 32, Lemma 33, Lemma 34, Lemma 35 and Lemma 36 since the value of V only has six possibilities, i.e., $\{p, d, i_p, i_d, i_{dp}, \text{na}\}$. \square

Evaluation to XACML Component.

Corollary 1. *Let $\Pi = \Pi_{\mathcal{Q}} \cup \Pi_{\text{XACML}}$ be a program obtained by merging Request transformation program $\Pi_{\mathcal{Q}}$ and all XACML components transformation programs Π_{XACML} . Let M be an answer set of Π . Then,*

$$\llbracket X \rrbracket(\mathcal{Q}) = V \quad \text{if and only if} \quad \text{val}(X, V) \in M$$

where X is an XACML component.

Proof. It follows from Prop. 2, Prop. 3, Prop. 4, Prop. 5, Prop. 6, Prop. 7 and Prop. 12. \square